В. Говорухин, Б. Цибулин

КОМПЬЮТЕР В МАТЕМАТИЧЕСКОМ ИССЛЕДОВАНИИ

Эта книга посвящена программным средствам, позволяющим провести весь цикл математического исследования: от непосредственного решения задачи (аналитического или численного) до подготовки статьи к печати. Под единой обложкой даны описание и примеры использования системы аналитических вычислений Maple, вычислительного пакета MATLAB и системы подготовки публикаций LaTeX.

В книге убедительно показывается, что система аналитических вычислений Maple и вычислительная среда MATLAB — хороший выбор для проведения любого исследования, где требуется математика — от курсовой работы до научного открытия.

Краткое содержание

Содержание

Алфавитный указатель

Символы

A

B

Q

R

S

Введение

Таких систем — пропасть. Но для эрцгерцога, наверно, купили что-нибудь этакое, особенное.

Йозеф Швейк

В настоящее время научное программирование претерпевает серьезную трансформацию: развиваются интегрированные среды, основанные на алгоритмических языках, и растет применение универсальных математических систем (Maple, Mathematica, MATLAB, MatCad и др.). Эти системы имеют дружественный интерфейс, реализуют множество стандартных и специальных математических операций, снабжены мощными графическими средствами и обладают собственными языками программирования. Все это предоставляет широкие возможности для эффективной работы специалистов разных профилей, о чем говорит активное применение математических пакетов в научных исследованиях и в преподавании. Эта книга — попытка объяснить, что система аналитических вычислений Maple и вычислительная среда MATLAB — хороший выбор для проведения любого исследования, где требуется математика — от курсовой работы до научного открытия. С помощью этих пакетов проще готовить и выполнять задания, устраивать демонстрации и гораздо быстрее решать исследовательские и инженерные задачи.

Конечным продуктом исследования выступают публикации, подготовка, распространение и использование которых в настоящее время требует квалифицированного применения компьютера. Это касается редактирования текста, изготовления графических материалов, ведения библиографии, размещения электронных версий в Интернете, поиска статей и их просмотра. Де-факто сейчас стандартными системами подготовки научно-технических публикаций являются различные реализации пакета TeX и текстовый редактор Word. Кроме того, необходимы минимальные знания о стандартных форматах файлов, конверторах, программах и утилитах, используемых при подготовке публикаций.

Данная книга посвящена программным средствам, позволяющим провести весь цикл математического исследования: от поиска и просмотра необходимой литературы до непосредственного решения задачи (аналитического и/или численно-

го) и подготовки статьи к печати. Под единой обложкой даны описание и примеры использования системы аналитических вычислений Maple, вычислительного пакета MATLAB, системы подготовки публикаций LaTeX. Выбор этих пакетов обусловлен их универсальными математическими возможностями, широкой распространенностью в России и за рубежом, а также взаимной интегрированностью. Для проведения аналитических преобразований в MATLAB используется Maple, а из Maple для численных расчетов можно обращаться к MATLAB, документы Maple автоматически преобразуются в документы LaTeX или HTML-страницы, а рисунки, полученные Maple и MATLAB, сохраняются практически во всех распространенных форматах.

Математические пакеты Maple и MATLAB – интеллектуальные лидеры в своих классах и образцы, определяющие развитие компьютерной математики. Компьютерная алгебра Maple вошла составной частью в ряд современных пакетов, численный анализ от MATLAB и наборы инструментов (Toolboxes) уникальны. Сами пакеты постоянно совершенствуются, развивая аппарат и пополняя ресурсы. Пакет Maple и вычислительная среда MATLAB – мощные и хорошо организованные системы, надежные и простые в работе. Освоение даже части их возможностей даст несомненный эффект, а по мере накопления опыта придет настоящая эффективность от взаимодействия с ними. Еще одним достоинством пакетов является неизменность набора основных команд и конструкций языка при появлении новых версий, и именно этим командам и конструкциям мы уделяем основное внимание в книге.

Сейчас пакеты Maple и MATLAB настолько велики, что изучение всех их возможностей может потребовать всей жизни. Пакеты достаточно быстро развиваются, и это заставляет вспомнить апорию Зенона об Ахиллесе, не могущем догнать черепаху. Фирмы Maple Software и Mathworks создают новые версии своих пакетов отнюдь не черепашьими темпами, а в качестве Ахиллеса приходится выступать многочисленным пользователям. Нам представляется, что Ахиллесу для успешного движения по дистанции нужен некоторый начальный уровень знаний, освоить который поможет настоящая книга, а далее каждый должен совершенствовать свои познания Maple и MATLAB, решая возникающие проблемы и обучаясь новому в процессе преодоления сложностей и трудностей.

Книга состоит из трех частей. Первые две части посвящены описанию основных возможностей пакетов MATLAB и Maple, их языкам и командам. Изложение сопровождается примерами использования команд и языковых конструкций. В третьей части изложены основы подготовки публикаций в стандарте LaTeX и даны сведения о форматах файлов, утилитах, средствах и ресурсах Интернета.

Первое знакомство с Maple и MATLAB

Сколь значителен спектр задач, которые решаются при помощи пакетов Maple и MATLAB, видно из следующего перечисления:

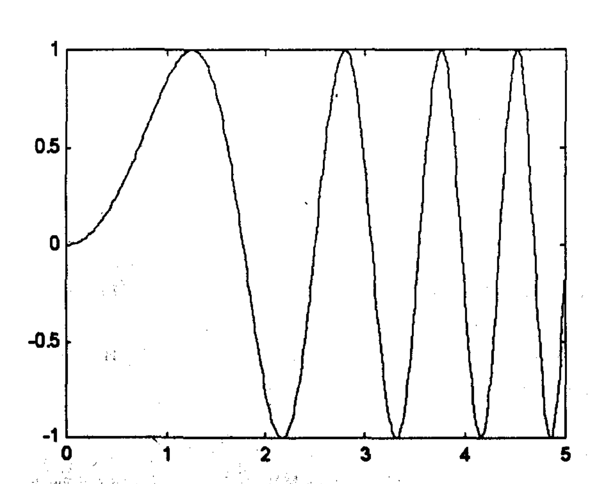
- **проведение математических исследований, требующих вычислений и аналитических выкладок;**

- разработка и анализ алгоритмов;
- математическое моделирование, компьютерный эксперимент;
- анализ и обработка данных;
- визуализация, научная и инженерная графика;
- разработка графических и расчетных приложений.

Оба пакета предоставляют удобные среды для компьютерных экспериментов, когда пробуются различные подходы к задаче, анализируются частные решения, при необходимости программирования отбираются требующие особой скорости фрагменты. Эти пакеты позволяют создавать интегрированные среды с участием систем программирования. Когда расчеты проведены и требуется оформить результаты, опять-таки можно использовать эти пакеты для визуализации данных и подготовки иллюстраций. Для завершения работы остается подготовить печатный материал (отчет, статью, книгу) и можно приступать к очередному исследованию. Работа с обеими системами проходит интерактивно — пользователь вводит команды и видит на экране результат их выполнения. Квалификация пакетов обеспечивает выбор подходящих типов переменных и выполнение операций, так что в общем случае не требуется описания переменных.

Основа MATLAB — это работа с матрицами, так что даже вычисления со скалярами реализуются как операции с матрицами размера 1×1. Матричные команды написаны особенно тщательно, и всюду, где это возможно, целесообразно пользоваться матричными (векторными) операциями, что ускоряет вычисления и предупреждает возможные ошибки. Матрицам, их свойствам и операциям с ними посвящено огромное количество литературы, например книга [46]. MATLAB в первую очередь предназначен для численного исследования, а потому оперирует в основном с числовой информацией. Приведем пример построения графика функции средствами MATLAB:

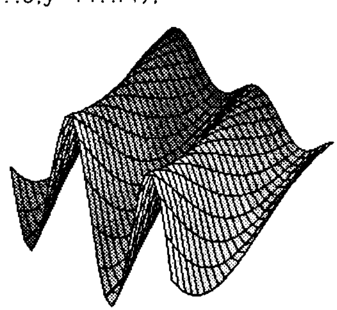
```
» x=[0:0.01:5];
» y=sin(x.^2);
» plot(x,y)
```



Для построения графика потребовалось определить значения аргумента в наборе точек (в переменной x содержатся точки 0, 0.01, 0.02, ..., 5) и вычислить значения функции в них. Для возведения в квадрат каждого числа из набора x использована так называемая поточечная операция (перед знаком операции стоит точка), так как обычное возведение в степень в MATLAB означает умножение матрицы на себя. Функция синуса была вычислена сразу для всех элементов переменной x.

Система Maple создавалась как пакет компьютерной алгебры, то есть основным объектом здесь являются формулы и операции с ними. Без дополнительных указаний символ, например x, считается фактически математической переменной, как x в формуле f(x). Такая специфика систем компьютерной алгебры позволяет проводить точные вычисления (см., например, [41]). Построим с помощью Maple поверхность, определяемую функцией двух переменных x и y. Для построения требуется указание интервала изменения x и y:

```
> plot3d(1.5^x*sin(2*y-x),x=-1..3,y=-Pi..Pi);
```



Проиллюстрируем работу двух систем на примере вычисления интеграла функции одной переменной. Сначала обратимся к команде интегрирования Maple:

```
> int((x-1)^2*sin(x),x);
```

$$-x^2 \cos(x) + \cos(x) + 2 \sin(x)\, x - 2 \sin(x) + 2\, x \cos(x)$$

Полученая формула — результат вычисления неопределенного интеграла. Если указать пределы интегрирования, то будет выведено значение определенного интеграла, причем в точном виде:

```
> res:=int((x-1)^2*sin(x),x=0..Pi);
```

$$res := -2 + \pi^2 - 2\pi$$

Чтобы получить ответ в виде числа, требуется обратиться к следующей команде:

```
> evalf(res);
```

1.586419096

В стандарте MATLAB можно получить только числовое значение, причем для этого нужно определить узлы для аппроксимации функции и значения функции в них, а затем обратиться к процедуре вычисления интеграла одним из методов (в примере — метод трапеций):

```
» x=[0:0.01:pi];
» y=(x-1).^2.*sin(x);
» res=trapz(x,y)
```

```
res =
  1.5864e+000
```

Отметим, что указанная специализация систем весьма условна, так как в состав MATLAB может входить библиотека компьютерной алгебры Symbolic, а последние версии Maple успешно справляются с численными расчетами.

Обе системы имеют развитые средства программирования, аналогичные распространенным алгоритмическим языкам, но при этом и здесь сохраняется специфика, обусловленная направленностью пакетов. В табл. 0.1 приведем примеры реализации одной конструкции средствами MATLAB, Maple и на алгоритмических языках Фортран и Паскаль.

Таблица 0.1. Сравнение четырех вычислительных сред

Язык	Конструкция
MATLAB	X=1:4; Y=sin(X)
Maple	x:=seq(k,k=1..4): y:=evalf(map(sin,[x]));
Фортран	REAL X(4), Y(4) DO 100 I=1,4 X(I)=I Y(I)=sin(Y(I)) 100 CONTINUE WRITE((6,'(4F10.4)'),Y)
Паскаль	var i:integer; X,Y : array [1..4] of real; Begin for i=1 to 4 do begin X[i]:=i; Y[i]:=sin(X(i)); write(' ',Y[i]) end; end

Из табл. 0.1 видна удивительная лаконичность языков MATLAB и Maple. Предварительного описания переменных не требуется, вектор формируется указанием диапазона чисел, а вычисление синуса для всех векторных элементов потребовало одной строки.

Наш подход при дальнейшем описании MATLAB и Maple состоит в том, чтобы дать как можно более широкое представление о возможностях систем, а перечисление и описание многочисленных вариантов применения всех команд можно найти в документации и системе справки пакета, а также в литературе (см. список в конце книги). Перед тем как перейти к описанию систем, коротко обсудим вопрос о решении задач с помощью компьютера.

Компьютерное исследование

Даже небольшая исследовательская или конструкторская проблема состоит из ряда этапов, для выполнения которых многократно приходится обращаться к помощи

компьютера. Постановка задачи и ее уточнение, анализ простейших моделей и ключевых факторов, пробное исследование, построение расчетной модели и обсчет задачи, обработка результатов и... И с высокой вероятностью исследователя ждет повторение данного цикла или некоторых его частей: постановка, анализ, исследование, обработка и т. д.

При математическом моделировании естественнонаучных и технических задач нужно быть готовыми к тому, что для проведения намечаемых работ окажется недостаточно имеющихся знаний и умений и понадобится изучить, разобрать, освоить что-то новое. Заранее неизвестно, какие трудности возникнут при решении задачи и какие средства помогут справиться с проблемой. Речь здесь, конечно, не идет о решении типовых проблем (задач с ответами), которые также приходится решать в этой жизни. Хотя и простые проблемы могут вызвать большие сложности.

Перед описанием пакетов Maple и MATLAB, следуя Хемингу [74], сформулируем несколько общих замечаний о решении задач с помощью вычислительной машины. Перед тем как приступить к компьютерному исследованию математической задачи, важно тщательно обдумать ряд вопросов:

- Что известно об исходной задаче? То есть какими свойствами она обладает и вся ли эта информация учитывается при решении задачи (например, если задача обладает симметрией, то она должна сохранятся и в аппроксимациях). Каковы входные данные, каков интервал их изменения и как эти изменения могут повлиять на ход решения? Что приблизительно мы хотим получить в результате решения, как должен выглядеть предполагаемый ответ?
- Как добиться результата? В первую очередь это включает выбор способа (аналитическое исследование или численный анализ) и методов решения задачи, необходимого инструмента (программного продукта). Нужно подумать о выборе наилучшего метода, то есть приводящего к верному результату за кратчайшее время. Выяснить, как будут проверяться полученные на каждом шаге решения результаты. Это касается как непосредственно программирования, так и корректности полученных величин.
- Сколько усилий потребует решение поставленной задачи? Под усилиями здесь понимаются количество необходимого времени для освоения пакета, программирования и отладки, затрат машинного времени на решение задачи. Всегда следует задумываться о том, когда будут получены окончательные результаты.

Рассматриваемые в книге системы могут сильно помочь в поиске ответов на перечисленные вопросы. Однако следует помнить, что для квалифицированного использования даже таких «умных» систем, как Maple и MATLAB необходимы хотя бы начальные знания из области математики, численного анализа и программирования. В конце книги дан список учебной и научной литературы, содержащей такую информацию.

Благодарности

Мы признательны фирмам Waterloo Maple Inc. и MathWorks, предоставившим нам последние реализации своих пакетов и ценные советы своих сотрудников. Спа-

сибо коллегам по работе и специальности за полезные предложения, ободрение и помощь. Особая благодарность редакции издательства «Питер» в лице Екатерины Строгановой и Андрея Васильева за терпеливое внимание и доброжелательную поддержку, без которых мы бы не закончили этой книги. Бесконечное спасибо нашим семьям за все.

Отдельная благодарность всем покупателям и читателям этой книги.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Подробную информацию о наших книгах вы найдете на web-сайте издательства http://www.piter.com.



Универсальный математический пакет Maple

- Основы Maple
- Аналитические преобразования в Maple
- Математический анализ в Maple
- Решение уравнений в Maple
- Алгебра в Maple
- Графика Maple
- Программирование в Maple
- Математические библиотеки Maple
- Maple и другие программы
- Примеры решения задач

В данной книге описывается система Maple 6, выпущенная фирмой Waterloo Maple Inc. в 2000 году. Приводимые примеры опробованы на IBM PC под управлением Windows 98 и, соответственно, изложение ведется применительно к реализации для графической оболочки Windows 98. Вместе с тем большая часть книги посвящена командам, языку и тем возможностям Maple, которые не зависят от типа используемой платформы.

Для установки Maple 6 нужен компьютер, имеющий 30 Мбайт на жестком диске и минимум 16 Мбайт оперативной памяти. Наличие лучшей машины, больших тактовой частоты и оперативной памяти повышает комфортность работы и дает возможность решать более сложные задачи.

Пакет Maple — интерактивная программа, позволяющая проводить аналитические выкладки и вычисления, снабженная средствами двумерной и трехмерной графики, имеющая мощный язык программирования и богатую библиотеку математических формул и сведений. Работа с Maple заключается в том, что пользователь вводит математические выражения и инструкции (команды), а система пытается их выполнить и представить ответ. Получив (или не получив) ответ, пользователь вводит новые инструкции и так далее — взаимодействие с пакетом происходит в диалоговом режиме. Благодаря собственному языку программирования высокого уровня введенные выражения и инструкции, а также результаты выполнения команд — формулы, графики, таблицы и числа — запоминаются в едином документе (worksheet). Это обеспечивает уникальную технологию работы, когда чуть ли не все этапы математического исследования можно отразить в одном документе, и итоговый документ становится (быть может, с минимальными дополнениями) научной статьей, разделом в учебнике, отчетом.

1 ГЛАВА Основы Maple

В данной главе рассмотрен ряд общих вопросов: интерфейс (системы меню, значков и справки), кратко описана организация документа Maple, изложены общие сведения об основных объектах (переменных, константах, выражениях) и синтаксисе, а также дан обзор базовых типов Maple и основных математических функций.

Работа с Maple и интерфейс

Графический интерфейс Maple аналогичен имеющемуся в системах редактирования и подготовки текста и использует обычные средства работы с файлами и редактирования (мышь и клавиатура). После запуска выполняемого модуля wmaple или xmaple в среде Unix появляется оболочка с новым документом (worksheet). Характерное окно приведено на рис. 1.1. В верхней части окна расположено меню (пункты File, Edit и т. д.), чуть ниже — строка значков Toolbar для ряда часто выполняемых операций, еще ниже — строка значков Context Bar, организующих представление данных в сеансе. Затем следует одно или несколько окон с документами, в которых размещаются формулы, рисунки, сопровождающий текст и др. В нижней части окна находится полоса Status line, которая содержит информацию о системе.

Работа в Maple проходит в режиме сессии (session) — пользователь вводит команды, математические выражения, процедуры, которые воспринимаются и интерпретируются Maple. Каждая команда должна завершаться точкой с запятой (;) или двоеточием (:). В первом случае в строке под предложением будет выведен результат исполнения команды или сообщение об ошибке, во втором случае результат не выводится. Для отмены всех сделанных назначений и начала нового сеанса без выхода из Maple используется команда restart.

Кроме того, в Maple можно вводить таблицы и текстовые параграфы, структурировать текст и документ, добавлять гиперссылки, объединяющие несколько документов в подобие электронной книги. В документ также можно вставлять объекты (рисунки и таблицы) из других программ, используя интерфейс OLE2.

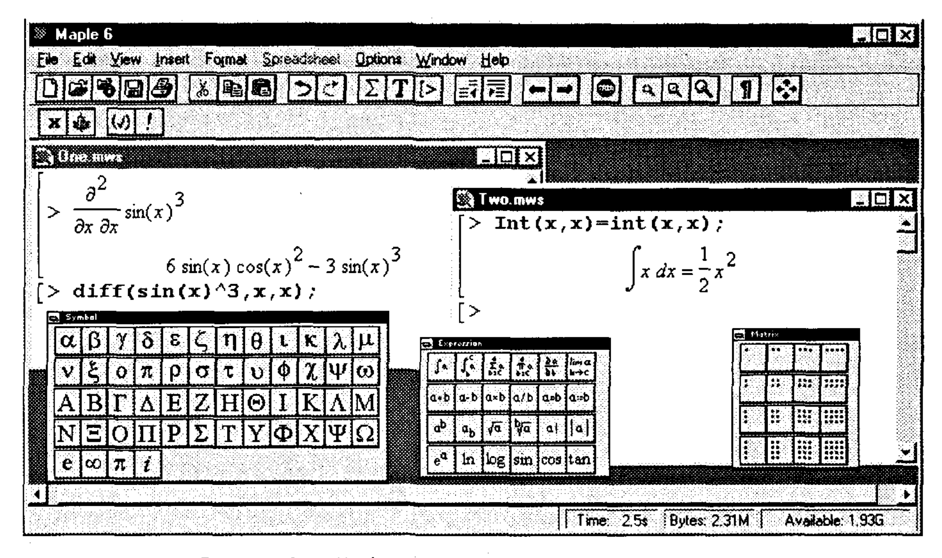


Рис. 1.1. Окно Maple с двумя документами и тремя палитрами

Отметим, что различные документы, открытые в одном сеансе, используют общую область памяти, и значение, присвоенное переменной в одном документе, сохраняется при переходе к другому документу. Для реализации многозадачной работы имеется программа Parallel Server Maple 6. Ее интерфейс идентичен основной программе wmaple, но при работе назначения переменных действительны только в пределах «родного» документа.

Оболочка

Команды Maple набираются после приглашения (>). Возможны два способа представления вводимого: стандартная математическая нотация и нотация Maple. В первом случае на экране дисплея интегралы, суммы и др. даются своими математическими изображениями, а во втором — при помощи текстовых эквивалентов. Например, на рис. 1.1 для документа One.mws использована математическая нотация, а для документа Two.mws — стандартная.

Нажатие клавиши Enter запускает исполнение введенных команд. Если интерпретатор посчитал введенное законченным предложением, то команды выполняются, в противном случае Maple ожидает завершения ввода. Обнаружив ошибку, Maple печатает на следующей строке сообщение о ней; при синтаксической ошибке символом ^ отмечается первая нераспознанная литера.

Результаты работы могут быть сохранены в файлах различных форматов. Текущий документ (области ввода и вывода, комментарии, текст, графика) записываются в файл с расширением .mws. При записи в файлы с другими расширениями сохраняются только области ввода и тексты комментариев. Кроме того, весь документ или его часть могут быть сохранены в форматах, допускающих их использование в других программах.

Пакет Maple постоянно развивается, приобретая новые команды и, соответственно, возможности для проведения математических выкладок, вычислений, графического отображения информации. При этом происходят изменения в синтаксисе и оболочке. Для сохранения преемственности с документами, подготовленными в старых версиях, фирма снабжает свой продукт специальными конверторами, при загрузке файла устаревшей версии Maple предлагает преобразовать нотацию файла согласно новым правилам.

Организация документа

Набор и исполнение команд производится по группам (Execution Group). Каждая группа состоит из нескольких частей: область ввода (Input Region) с командами, область вывода (Output Region) с результатами выполнения и тексты комментариев (Text Region). Область вывода может включать результаты выполнения математических и алгоритмических операций, а также графические образы (двумерная и трехмерная графика). Группы выделяются слева квадратными скобками и разделяются сепараторами (Separator), которые по умолчанию невидимы, но могут быть сделаны видимыми. Для структурирования документа используются параграфы.

Система меню

Основное меню состоит из пунктов, приведенных в табл. 1.1.

Таблица 1.1. Основное меню

Пункт	Назначение
File	Работа с файлами, печать
Edit	Правка
View	Просмотр
Insert	Вставка
Format	Форматирование документа
Spreadsheet	Работа с таблицами
Options	Установки и параметры
Window	Работа с окнами
Help	Справка

Представим отдельные меню Maple (см. табл. 1.2–1.9), группируя близкие по смыслу пункты. Многоточие в пунктах меню означает появление дополнительного окна для указания имени читаемого документа, вызываемой страницы URL в Интернете.

Знак многоточия после имени команды указывает на то, что при выборе данного пункта появится дополнительное окно (например, для выбора файла), а знак ▸ после пункта меню указывает на дополнительное меню. Так, при выборе пункта Export As вызывается меню форматов для экспорта документа (HTML..., LaTeX..., Maple Explorer..., Maple Text..., Maple Plain..., RTF...).

Таблица 1.2. Меню File — работа с файлами, печать

Пункты	Назначение
New	Создать новый документ
Open.../ Open URL...,	Открыть документ / открыть страницы URL
Save.../ Save As	Сохранить / Сохранить как
Export As	Экспортировать в виде
Close	Закрыть документ
Save Settings / AutoSave Settings	Сохранить параметры Maple / режим автоматического сохранения
Print / Print Preview / Print Setup	Печать / просмотр страниц для печати / установка параметров печати
Exit	Выход

Назначение пунктов меню правки достаточно традиционное и в основном соответствует стандартам Windows (см. табл. 1.3). Копирование текста Maple и соответствующая вставка предназначены для того, чтобы при операциях опустить вывод, а перенести только команды Maple.

Таблица 1.3. Меню правки Edit

Пункты	Назначение
Undo / Redo	Отменить операцию, вернуть операцию
Cut	Вырезать в буфер
Copy / Copy as Maple Text	Копировать в буфер / Копировать как текст Maple
Paste / Paste Maple Text	Вставить из буфера / Вставить как текст Maple
Delete Paragraph	Удалить параграф
Select All	Выделить все
Find	Поиск и замена
Hyperlinks	Редактирование гиперссылок
Object	Редактирование объекта OLE 2
Entry mode	Переключение моды ввода от стандартной математической к текстовой
Split / Join	Разделение и объединение групп и секций
Execute	Вычисление выделенной части или всего документа
Remove Output	Удаление области вывода для выделенной части или всего документа

Внешний вид оболочки, размер символов, оформление групп определяются при помощи меню **View** (см. табл. 1.4).

При выборе пункта **Palettes ▸** появляется меню, позволяющее по выбору установить палитры греческих букв (**Symbol Palette**), шаблонов математических выражений (**Expression Palette**) и задания матриц (**Matrix Palette**), задать режим присутствия (**Show All Palettes**) или отсутствия (**Hide All Palettes**) всех палитр.

Таблица 1.4. Меню View — вид оболочки

Пункты	Назначение
Status Bar Toolbar Context Bar	Переключатели, регулирующие отображение строки состояния, а также панелей со значками операций и оформления
Palettes	Вспомогательные палитры греческих букв, математических шаблонов и матриц
Zoom Factor	Выбор разрешения (50–400%)
Bookmarks	Редактирование закладок
Back / Forward	Перемещение назад / вперед по гиперссылкам
Hide Content	Подменю переключателей, позволяющих спрятать таблицы, ввод, вывод, графику
Show Invisible Characters, Show Section Ranges, Show Group Ranges, Show OLE type	Переключатели, регулирующие отображение служебных символов, диапазонов секций, диапазонов групп и объектов OLE
Expand All Sections / Collapse All Sections	Раскрытие и схлопывание секций

В режиме отображения диапазонов секций (групп) устанавливаются вертикальные линии для отметки секций (групп). Возможность одновременно раскрыть или закрыть все секции облегчает работу с большими структурированными документами.

Назначение пунктов меню вставки (см. табл. 1.5) естественно: осуществлять переход от текстового режима к режиму ввода команд и математических формул, используя различные нотации, вставлять группы, параграфы, разделы, подразделы и другое, то есть создавать документ, удобный для чтения и использования в научной работе, преподавании, публикациях и презентациях.

Таблица 1.5. Меню вставки Insert

Пункты	Назначение
Text / Standard Math	Переход в текстовый режим / переход в режим набора математической формулы в тексте
Maple Input / Standard Math Input	Переход в режим ввода команд / переход в режим набора команд в математической моде
Execution Group	Вставить группу
Plot	Вставить двумерный или трехмерный рисунок
Spreadsheet	Вставить таблицу
Paragraph	Вставить параграф (абзац)
Section, Subsection	Вставить раздел или подраздел
HyperLink...	Вставить ссылку на URL, документ или справку
OLE Object	Вставить объект OLE 2.0
Page Break	Вставить разделитель страниц

Подменю, появляющиеся по командам Execution Group и Paragraph, позволяют произвести вставку новой группы и параграфа до или после группы, отмеченной курсором. Для вставки группы после курсора имеется соответствующий значок, см. рис. 1.6.

При помощи подменю Plot можно вставить пустой двумерный или трехмерный график, то есть оси координат. Затем можно выделить в документе и скопировать (перетащить) в область рисунка различные функции или выражения для построения графика.

Текстовое оформление документа, параметры страницы, назначение стилей, выделение и используемые стили — все это устанавливается в меню Format (см. табл. 1.6).

Таблица 1.6. Меню Format — форматирование документа

Пункты	Назначение
Styles...	Редактирование стилей
Page Numbers...	Нумерация страниц
Italic, Bold, Underline	Выделение шрифта соответствующим стилем
Left Justify, Center, Right Justify	Режимы выравнивания
Paragraph, Character	Оформление текста, назначение шрифта
Indent, Outdent	Формирование секции, обратное преобразование
Convert to...	Преобразование выделенного текста в гиперссылку, математическую или Maple-нотацию

По команде Styles меню Format вызывается специальное окно для изменения стилей, см. рис. 1.2.

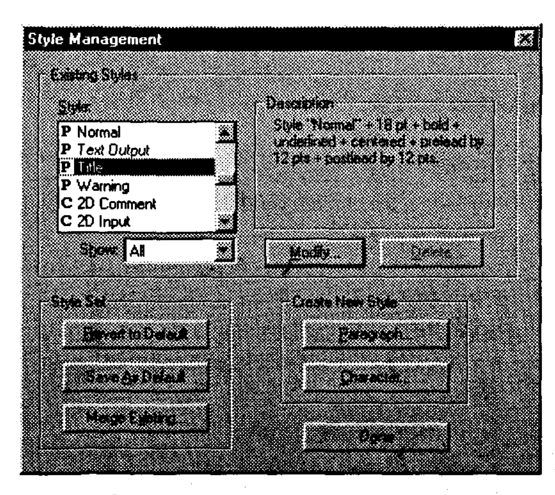


Рис. 1.2. Окно редактирования стилей

Нажатие кнопки Modify открывает окно изменения свойств выбранного стиля, например, рис. 1.3 соответствует стилю Maple Output.

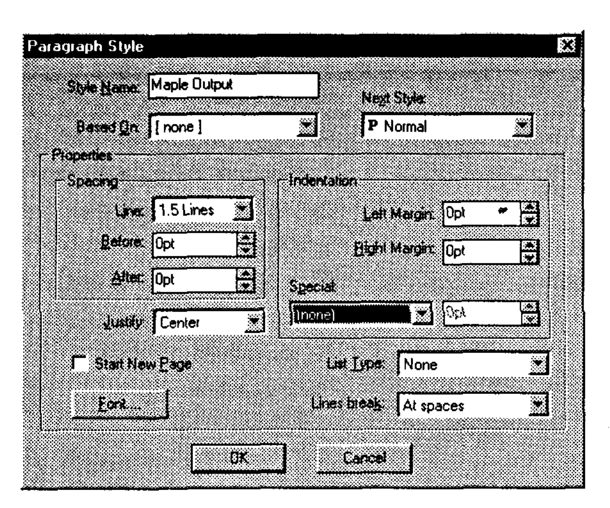


Рис. 1.3. Окно изменения свойств стиля

Для изменения свойств шрифта нужно нажать кнопку Font, в результате появится окно, приведенное на рис. 1.4.

Рис. 1.4. Свойства шрифта

Пункт меню Convert to... позволяет преобразовать текст к одному из следующих видов: текст (Text), формула в тексте (Standard Math), выражение, команда нотации Maple (Maple Input), выражение, команда в математической нотации (Standard Math Input).

Для активизации меню Spreadsheet (см. табл. 1.7) нужно выполнить команду Insert ▸ Spreadsheet. В документе при этом появляется таблица, а на панели Context Bar отображаются кнопки команд работы с таблицей. Способ взаимодействия с таблицами в Maple напоминает работу с другими электронными таблицами, в частности Excel.

Таблица 1.7. Меню Spreadsheet — работа с таблицами

Пункты	Назначение
Evaluate Selection Evaluate Spreadsheet	Вычислить выделенное / таблицу
Row	Размер, вставка, удаление строки
Column	Размер, вставка, удаление столбца
Fill	Заполнение выделенного
Import Data / Export Data	Считывание / сохранение данных
Properties	Свойства ячеек (точность, режим вычисления)
Show Border	Показывать обрамление
Resize to Grid	Масштабирование

Для создания таблицы нужно ввести команду Insert ▸ Spreadsheet. При этом появится таблица (см. рис. 1.5). Работа с таблицами может проводиться как интерактивно, так и с использованием пакета Spread (см. главу 8 «Математические библиотеки Maple»).

Установить необходимые свойства документа и параметры работы позволяют пункты меню Options (см. табл. 1.8). Это в основном переключатели.

Различные возможности расположения нескольких документов предоставляет меню Windows (см. табл. 1.9). Кроме того, это меню содержит имена открытых в сеансе документов и страниц справки, что позволяет переходить к нужному документу, если он скрыт за другими.

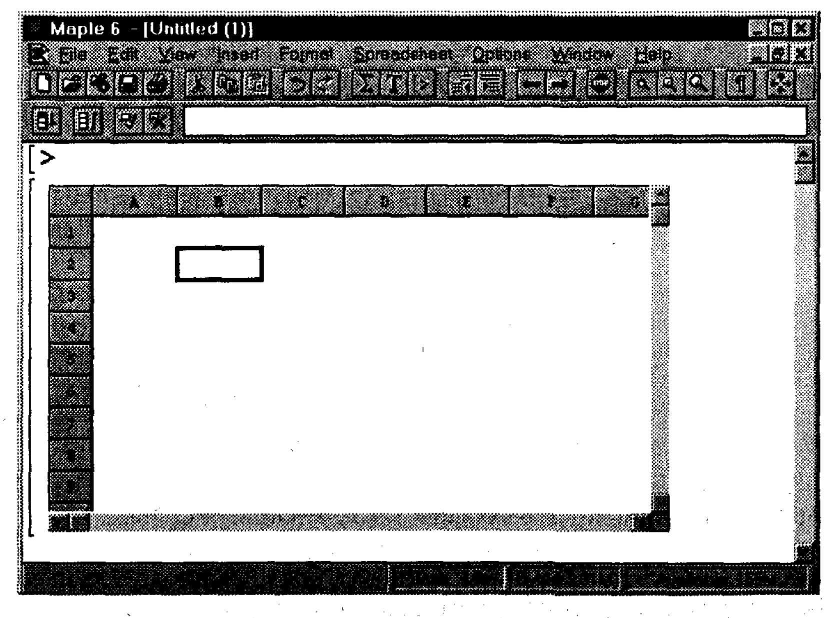


Рис. 1.5. Окно Maple при работе с таблицей

Таблица 1.8. Меню Options — возможности

Пункты	Назначение
Replace Output	Режим замещения при выводе (переключатель)
Insert Mode	Вставка новой группы при выполнении (переключатель)
Browser	Выбор браузера для открытия URL
Accept Launches	Запрос на выполнение команд Maple из других приложений
Export	Параметры экспорта документа (worksheet)
Input Display	Нотация для вводимых команд (стандартная или математическая)
Output Display	Нотация для вывода результатов
Assumed Variables	Управление видом переменных, относительно которых сделаны назначения
Plot Display	Режим вывода графики (документ или отдельное окно)
Display 2-D Legends	Вывод легенды для двумерных графиков
Print Quality	Качество печати
Palette Size	Размер палитр
Auto Save	Режим автосохранения

Таблица 1.9. Меню Windows — работа с окнами

Пункты	Назначение
Cascade / Tile	Расположить документы каскадом / без наложения
Horizontal / Vertical	Расположить документы в ряд по горизонтали / вертикали
Arrange Icons	Распределить значки
Close All / Close All Help	Закрыть все документы / Закрыть все справки

По команде Закрыть все Maple поинтересуется о сохранении тех документов, изменения в которые не были записаны, так что не следует опасаться потери информации при запуске этой и других команд. Не надо только отказываться или соглашаться не глядя.

Последний пункт меню — справка Help рассмотрен далее.

Значки и контекстное меню

Общепринято, что для ряда часто используемых в работе пунктов меню удобно ввести значки (иконки). В Maple помимо стандартных значков для работы с файлами и редактирования имеется набор специфичных значков. На рис. 1.6 даны пояснения для значков, относящихся к панели Toolbar.

В режиме ввода команд имеются четыре значка панели Context Bar, см. рис. 1.7.

В режиме ввода текстовой информации вместо строки с четырьмя кнопками появляется строка, где организован доступ к стилям документа, шрифтам и их размеру, см. рис. 1.8.

При выделении рисунка система значков меняется так, чтобы поддержать средства графического оформления рисунка, об этом подробно в главе 6 «Графика Maple».

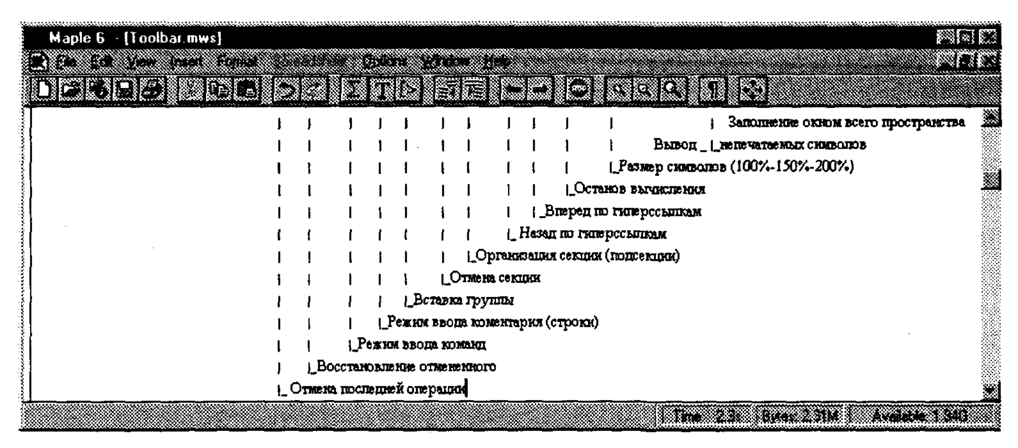


Рис. 1.6. Назначение ряда значков панели Toolbar

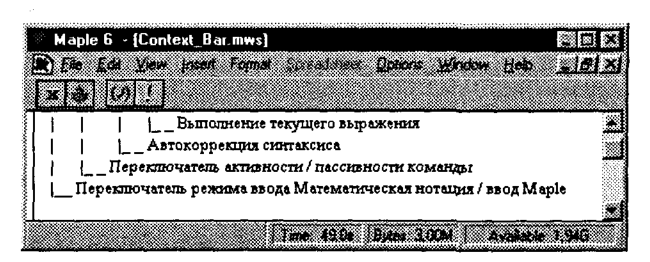


Рис. 1.7. Значки Context Bar в режиме ввода команд

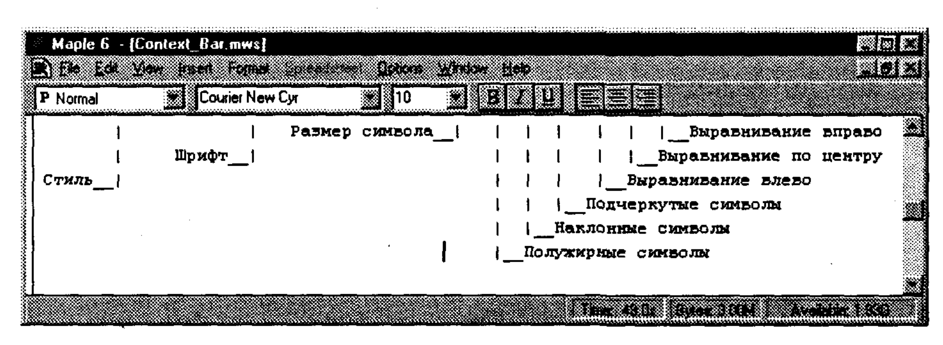


Рис. 1.8. Значки Context Bar в режиме ввода текста

В Maple 6 по сравнению с Maple V.5 существенно расширились возможности, предоставляемые правой кнопке мыши. В зависимости от контекста можно сделать текущую строку текстовой или командной, перейти от стандартной моды набора команд к математической, выделить группу строк и выполнить их, не производя построчное нажатие (**Execute**).

При указании на строку ввода появляется простое меню, которое позволяет преобразовать ее к математическому виду — **Standart Math** (или к стандарту Maple), сделать данную строку невыполнимой (или наоборот — **Maple Input**), выполнить команды на строке — **Execute**.

При указании области вывода набор возможностей контекстного меню гораздо шире. В зависимости от контекста может появиться либо простое меню для копирования результата в буфер, либо предложение преобразовать вывод в список (list), множество (set) или систему уравнений с нулевой правой частью. Возможно также появление меню, приведенного на рис. 1.9, которое кроме копирования предложит на выбор продифференцировать, проинтегрировать, разложить на множители, аппроксимировать, решить уравнение, выделить коэффициенты, сгруппировать по степеням, применить операции комплексной арифметики, сконструировать из выделенного сумму, произведение и т. д., конвертировать в стандарт LaTeX, Fortran и другие, отсортировать и нарисовать графики. При выборе пункта соответствующее действие будет произведено и результат появится на строке ввода специально организованной группы.

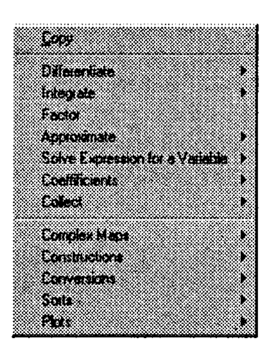


Рис. 1.9. Пример контекстного меню при указании области вывода

Нужно отметить, что пользователь может развить систему контекстного меню, добавив собственные пункты при помощи пакета context, см. справку Maple.

Справочная система

В Maple хорошо продумана организация справочной системы (см. рис. 1.10). Получить справку можно, обратившись к пунктам меню Help, или непосредственно из командной строки. Для пояснения значков и пунктов меню имеется режим всплывающей подсказки (Balloon Help).

В системе справки команды сгруппированы по разделам «Математика», «Графика», «Программирование» и так далее, каждый из которых в свою очередь состоит из набора тем. Например, математический раздел представлен темами «Алгебра», «Анализ» (Calculus), «Дифференциальные уравнения» и другими. Так, на рис. 1.10 выбраны раздел «Математика», подраздел «Алгебра», тема «Преобразования», группа команд раскрытия скобок и команда Expand. Справка для каждой команды состоит из описания команды и ее параметров, почти всегда содержит примеры и, кроме того, указывает на родственные команды. Такая организация справки, расширяя знания о системе Maple, позволяет использовать помещенные примеры в качестве прототипов и пробовать иные средства для проведения преобразований и вычислений.

Помимо структурированной информации можно использовать прямой поиск, чтобы ускорить получение информации по нужной команде или теме. При вызове

пункта Topic Search появляется окно (см. рис. 1.11), где можно ввести интересующее имя или его часть и получить список команд с именами, начинающимися с комбинации букв, помещенной в окошке Topic. На рис. 1.11 выведен ряд тем, названия которых начинаются с сочетания «inte», в том числе описание типа integer, ссылка на команды интегрирования, интерполирования и команду interface.

Рис. 1.10. Окно справки

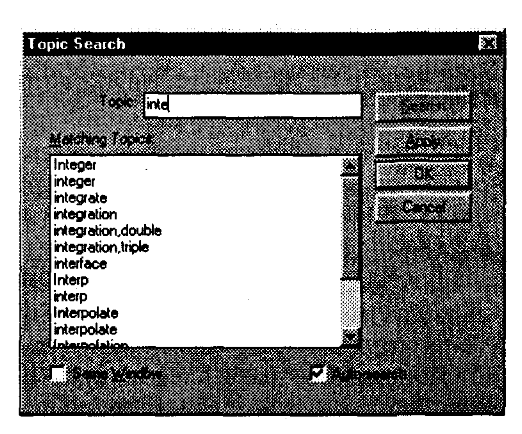


Рис. 1.11. Поиск по темам

Для проведения поиска по ключевому слову имеется система полнотекстового поиска, вызываемая пунктом меню Full Topic Search. Появляющееся при этом окно подобно рассмотренному выше для пункта Topic Search.

Для получения справки из командной строки используется специальный формат: в строке ввода набираются вопросительный знак и имя команды. Например,

запрос о пакете, команде или служебном слове topic выглядит следующим образом:

```
?topic
```

Заметим, что в конце строки отсутствует разделитель (двоеточие или точка с запятой), которым завершается каждая команда. Для получения информации о команде subtopic из пакета topic соответствующий запрос оформляется при помощи команд

```
?topic,subtopic
?topic[subtopic]
```

Получить справку из командной строки можно также при помощи команды help:

```
help(topic);
help(topic,subtopic);
help(topic[subtopic]);
```

Если при наборе имени команды, о которой запрашивается информация, была сделана ошибка, то Maple выведет перечень имен, которые могут иметь отношение к запросу. Это может быть полезным, если вы забыли точное имя команды, а также при интуитивном способе освоения Maple, когда пользователь предполагает, что для нужной операции может существовать команда с некоторым именем.

Для вывода информации о формате команды topic используется команда ??topic или ее аналог usage(topic). Для получения примеров из справки для команды topic можно выполнить команду ???topic или example(topic). Наконец, сведения о родственных для topic командах появятся в области вывода после запроса related (topic).

Вообще, начиная работу с Maple, полезно справиться о последних новостях реализации, которые могли быть не отмечены в печатных руководствах:

```
?updates
```

Конечно, данное представление пунктов меню не является исчерпывающе полным, но, мы предполагаем, что читателю по силам провести ряд экспериментов и установить не отмеченные в данном изложении возможности оболочки. По нашему мнению, для нормальной работы требуется ограниченное число интерфейсных средств, а мощность универсального математического пакета в первую очередь определяется начинкой — командами и библиотеками — тем аппаратом, который позволяет решать математические задачи.

Основные объекты

Системы аналитических вычислений оперируют со множеством объектов, используя для работы различные типы данных. Это позволяет применять свои правила обработки для каждого типа.

Простейшими объектами в Maple являются числа, константы, строки и имена. Под числами будем понимать собственно целые и вещественные числа, а также рациональные дроби и радикалы (корни из чисел).

Приведем примеры чисел:

```
> 1.23, 4e5,I+Pi, sqrt(8);
```

1.23, 400000., $I+\pi, 2\sqrt{2}$

Использование рациональных чисел, радикалов и констант (число π, мнимая единица) позволяет проводить абсолютно точные вычисления, так как при выполнении операций не возникает погрешности округления. Сразу отметим, что эти операции производятся гораздо медленнее.

Синтаксис и выражения

Каждая введенная команда должна завершаться разделителем: точкой с запятой (;) или двоеточием (:). Если ввод предложения завершается точкой с запятой, то в строке под предложением сразу будет отклик: результат исполнения команды или сообщение об ошибке. Разделитель (:) используется для отмены вывода, когда команда выполняется системой, но ее результат не выводится на экран.

В Maple применяются круглые, квадратные и фигурные скобки. Назначение круглых скобок — задавать порядок при построении математических выражений и обрамлять аргументы функций и параметры в записи команд. Квадратные скобки нужны для работы с индексными величинами. Фигурные скобки используются для формирования множеств.

В Maple две последовательные точки в параметрах команд применяются для определения интервала изменения переменных.

Теперь расскажем о других важных символах, используемых в Maple. Знаком процента (%) обозначается предшествующий вывод. Два знака процента отсылают к предпоследнему результату. Наконец, предшественник предпоследнего результата обозначается тремя знаками процента. Эта нотация может быть удобна при последовательной работе с документом, но чревата неприятностями при свободном перемещении по тексту, когда команды выполняются в произвольном порядке.

Используя константы, переменные, знаки арифметических и других операций, составляются выражения. Это основной объект для многих команд.

Последовательность выполнения арифметических операций соответствует стандартным математическим правилам: сначала проводится возведение в степень (^), затем умножение (*) и деление (/), а в конце — сложение (+) и вычитание (–). Операции выполняются слева направо, для изменения порядка используются круглые скобки.

Для операций отношения имеются знаки >, <, >=, <=, <>, =, а для конструирования булевых выражений используются также команды not or, and.

Обратный слеш (\) используется для переносов, а для комментирования в Maple предусмотрен символ #. Вся строка после этого символа не выполняется. Приведем примеры выражений:

```
> x+y^z-12<0;
```

$x+y^z<12$

```
3.14159\26535\89793\23846; # Число пи
```

3.14159265358979323846

Знак равенства (=) используется при формировании уравнений, а знак присвоения (:=) – при задании значений переменных. Различие в их использовании можно проиллюстрировать следующим примером:

```
> eq:=x=12; x;
```

$eq := x = 12$

x

Переменной eq присвоено уравнение x=12, но это не означает, что переменная x получила значение 12. Для того чтобы значением переменной x стало число 12, нужно ей присвоить это значение:

```
> x:=12;
```

$x := 12$

Константы

В Maple представлены все основные математические константы. В табл. 1.10 перечислены важнейшие из них. Напомним, что число π дается при помощи Pi, а pi означает греческую букву π.

Таблица 1.10. Математические константы

Имя	Описание
Pi	Число π
E	Основание натурального логарифма
I	Мнимая единица (квадратный корень из −1)
Infinity	Бесконечность
gamma	Константа Эйлера
true, false	Булевы константы (истина, ложь)

Имена этих констант являются зарезервированными, а их значения не могут быть переопределены в отличие от ряда переменных среды (управляющих констант).

Переменные

Переменная Maple идентифицируется именем — набором символов, начинающимся с буквы, причем большие и малые буквы различаются. Кроме букв могут употребляться цифры и знак подчеркивания. Приведем примеры различных имен:

```
Old, old, o_ld, Old
```

Длина имени зависит от платформы, и на 32-битных машинах допустимы имена из пятисот тысяч символов, а для 64-битных машин можно составлять имена длиной более миллиарда знаков (см. справку). Составные имена могут быть сформированы при помощи оператора конкатенации (||) или команды cat.

В качестве имен переменных запрещено использовать термины языка Maple:

```
and, break, by, catch, description, do, done, elif, else, end, error, export, fi, finally,
for, from, global, if, in, intersect, local, minus, mod, module, next, not, od, option,
options, or, proc, quit, read, return, save, stop, then, to, try, union, use, while
```

Кроме того, не рекомендуется использовать имена всех команд Maple в качестве имен.

Для обозначения служебных констант используются имена, начинающиеся с подчеркивания. Неопределенные константы, возникающие при решении дифференциального уравнения, именуются _C1, _C2 и т. д. Произвольное целое число обозначается как _N1, _N2 и т. д., а комплексная величина соответственно как _Z1, _Z2 и т. д. Если последним символом имени идет тильда (~), то это имя переменной, относительно которой сделаны назначения (определена вещественность или положительность и т. д., см. подробнее команду assume).

Для присвоения значений переменной используется знак :=. Для просмотра содержимого переменной простого типа нужно лишь ввести имя переменной (для массивов и других составных типов используется команда eval). Напомним, что команда restart используется для отмены всех сделанных назначений в сеансе. Чтобы освободить конкретную переменную от предшествующих назначений, нужно этой переменной присвоить ее имя, заключенное в прямые одинарные кавычки (') (апострофы). Например:

```
> ex:=x^2+exp(y): ex;
```

$x^2 + e^y$

```
> ex:='ex': ex;
```

ex

Для защиты от изменений существует команда protect, а для снятия защиты — unprotect. В частности, защищенными являются многие константы Maple. Приведем пример использования последних двух команд. Сначала переменной присвоим значение, а затем ее защитим:

```
> a:=2: protect("a"): a;
```

2

Теперь попытаемся присвоить ей иное значение. Результатом будет сообщение об ошибке:

```
> a:=3;
Error, attempting to assign to "a" which is protected
```

Следующей командой снимем защиту с переменной, а затем присвоим ей новое значение:

```
> unprotect("a");
> a:=3: a;
```

3

Переменные среды

Важными переменными среды являются Digits и Order, определяющие соответственно число знаков мантиссы для операций с плавающей запятой (по умолча-

нию десять цифр) и порядка разложений (по умолчанию разложения выписываются члены до шестого порядка). Для переопределения любой из этих величин достаточно просто присвоить ей новое значение.

Приведем примеры, иллюстрирующие потерю точности при вычислениях с недостаточным числом значащих цифр и вычисление экспоненты, напоминающее о годе рождения «зеркала русской революции» (Л. Н. Толстой, 1828 г.):

```
> Digits:=1: sqrt(2.0)+0.01-71/50;
```

0.

```
> Digits:=4: sqrt(2.0)+0.01-71/50;
```

.004

```
> Digits:=20: exp(1.0);
```

2.7182818284590452354

Операции с вещественными числами проводятся по умолчанию с десятью значащими цифрами, но, переопределив `Digits`, можно работать с любой мантиссой. Этот способ может оказаться полезным и для определенных этапов символьных вычислений, поскольку операции с рациональными числами выполняются медленнее.

Имеется также ряд других переменных среды. Так, переменная среды `UseHardwareFloats` определяет использование арифметического процессора компьютера для операций с вещественными числами, на котором вы работаете. Если эта переменная имеет значение `true`, то используется непосредственно процессор, а в случае `false` арифметические операции проделываются Maple программно. По умолчанию этой переменной присвоено значение `true`. В настоящее время этот режим действует для команд пакета **LinearAlgebra** и обслуживает операции с массивами, матрицами и векторами, которые основаны на новом классе `rtable`. Со временем, по утверждению разработчиков, переменная `UseHardwareFloats` будет определять режим вычислений для всех операций арифметики с плавающей точкой.

Перечень переменных среды может быть выведен по команде

```
> anames(environment);
```

Testzero, *UseHardwareFloats*, *Rounding*, *%*, *%%%*, *Digits*, *index/newtable*, *mod*, *%%*, *Order*, *printlevel*, *Normalizer*, *NumericEventHandlers*

Строки и символы

Строкой (`string`) является любой набор символов, заключенный в двойные кавычки. Например:

```
> "Maple string:"";'<> ?,./'{}[]`~!@#$%^&*()_+";
```

"Maple string:\";'<> ?,./'{}[]`~!@#$%^&*()_+

Отметим, что для включения двойных кавычек их нужно продублировать, при этом их предваряет обратный слеш. Так, результатом команды из предыдущего примера является одна строка. Строка отличается от символа, который получается, если фразу заключить в обратные кавычки:

```
> `This is a Maple symbol`;
```

This is a Maple symbol

Символ воспринимается Maple как единое целое, а строка состоит из символов, и с каждым из них можно работать отдельно. Например:

```
> v1:="String"; v2:="Symbol";
```

v1 := "String"

v2 := *Symbol*

```
> v1[2]; v2[1];
```

"t"

$Symbol_1$

Команды

Выражения и переменные обычно служат параметрами команд Maple. Стандартное обращение к команде command выглядит следующим образом:

```
command(par1,par2,...);
```

Здесь command — имя команды, а par1, par2, ... — ее параметры. Результат выполнения команды может быть присвоен некоторой переменной.

Наиболее важные команды содержатся в ядре Maple и вызываются автоматически, команды из главной библиотеки загружаются в память при их вызове. Остальные команды являются частью пакетов (библиотек). До запуска таких команд пакет должен быть загружен командой

```
with(package)
```

Здесь package — имя пакета. Такими пакетами являются: DEtools, Domains, GF, GaussInt, Groebner, LREtools, LinearAlgebra, Matlab, Ore_algebra, PDEtools, Slode, Spread, algcurves, codegen, combinat, combstruct, context, diffalg, difforms, finance, genfunc, geom3d, geometry, group, inttrans, liesymm, linalg, networks, numapprox, numtheory, orthpoly, padic, plots, plottools, polytools, powseries, process, simplex, stats, student, sumtools, tensor. Назначение каждого пакета описано в главе 8 «Математические библиотеки Maple», где указаны главы, в которых рассмотрены команды данного пакета.

Если из пакета package нужна одна команда command, то можно загрузить только ее:

```
with(package,command),
```

Можно также использовать вызов команды с префиксом пакета:

```
package[command](par1,par2,...).
```

Пакеты статистики stats и интегрирования дифференциальных уравнений DEtools, в свою очередь, состоят из пакетов, способы обращения с которыми можно узнать, обратившись, например, к документации по пакету stats.

При подключении ряда пакетов происходит переопределение некоторых команд. Так, команда трассировки trace после подключения пакета линейной алгебры linalg замещается командой вычисления следа матрицы. Предупреждение о переопределенных командах выводится при загрузке пакета.

Возможные ошибки

Если команда введена правильно и полностью, то Maple выполняет ее и приводит в следующих строках результат. Если появилось эхо введенной команды или об-

ласть вывода пуста, то либо Maple отказывается выполнить команду из-за неполноты информации, либо не может ее выполнить (уравнение не решается, интеграл не берется и т. п.). В этом случае полезно задуматься о том, что делается: подключена ли нужная библиотека, решается ли в принципе поставленная задача, нет ли других подходов, методов, команд.

Предупредим об опасности произвольного назначения имен переменных. Если имя переменной совпадает с именем какой-нибудь команды, то такая команда становится недоступной в текущем сеансе. Например:

```
> dchange:=2;
```

dchange := 2

```
> with(PDEtools,dchange);
Error, (in pacman:-with) pacman:-pmember expects its 1st argument, nom, to be of type name, but received 2
```

Поэтому перед введением новой переменной name полезно удостовериться, что имя не занято, командой

```
?name
```

Появление окошка справки будет означать существование команды с именем name.

При работе с Maple следует анализировать результат выполнения команды. Будучи гибкой системой, Maple старается выполнить любую введенную команду. Синтаксические ошибки выделяются программой и достаточно заметны. Описки не так бросаются в глаза. Если неправильно введено имя команды или не загружена библиотека, то Maple просто повторяет набранное в строке ввода без выполнения. Вообще, нужно постоянно проверять введенное и результат, иначе могут возникать «непонятные» ошибки. Например, если в строке ввода опустить знак умножения перед круглой скобкой в выражении x*(y-z), то Maple посчитает, что y-z является аргументом некоторой функции x(y-z). Сказанное иллюстрирует пример дифференцирования двух этих функций по переменной y:

```
> diff(x*(y-z),y);
```

x

```
> diff(x(y-z),y);
```

$\mathrm{D}(x)(y-z)$

Другая часто встречающаяся ошибка связана с тем, что Maple различает большие и маленькие буквы. Например, если имя числа π написать с маленькой буквы, то пользователь не заметит видимых изменений при представлении результата команд, но многие команды будут работать не верно. Пример:

```
> sin(pi);
```

$\sin(\pi)$

```
> sin(Pi);
```

0

Как уже отмечалось выше, прямые одинарные кавычки (') (апостроф) используются для того, чтобы освободить переменную от предшествующих назначений. Кроме того, прямые кавычки полезны для предупреждения ошибки в том случае, когда для выполнения команды используется переменная, получившая значение

ранее. Выполним последовательно команду присвоения и команду вычисления суммы, использовав в качестве индекса определенную ранее переменную:

```
> i:=3;
```

$i := 3$

```
> sum(i^2,i=1..6);
Error, (in sum) summation variable previously assigned, second argument evaluates to 3 = 1
.. 6
```

В результате появилось сообщение об ошибке, поскольку второй аргумент в команде суммирования `sum` воспринят как `3=1..6`. Применение кавычек даст правильный результат:

```
> sum("i^2",'i'=1..6);
```

91

Использование системы аналитических вычислений не отменяет мыслительного процесса, а только помогает ему.

Типы переменных

В Maple существует множество типов переменных: от известных вещественного (`float`), целого (`integer`) и строкового (`string`) до тех, которые необходимы для выполнения и программирования аналитических преобразований: дробь (`fraction`), функция (`function`), индексная переменная (`indexed`), процедура (`procedure`), множество (`set`), разложение (`series`), последовательность выражений (`exprseq`), массив (`array`), списки (переменные типа `list`, `listlist`, `listlistlist`) и некоторые другие. Списки используются для хранения коэффициентов полиномов, множества удобны при формировании системы уравнений и решений уравнений, массивы позволяют хранить и использовать упорядоченную информацию и т. д.

По умолчанию переменная считается скалярной и имеет тип `string`. Это фактически математическая переменная, как `x` в формуле `f(x)`. Для задания переменных других типов требуется явное их определение: при помощи оператора присваивания или команд, преобразующих тип.

Информацию о типе той или иной переменной можно получить при помощи команды `whattype`, а проверить принадлежность переменной `VAR` типу `TYP` помогает команда

```
type(VAR,TYP);
```

Рассмотрим базовые типы: последовательности выражений, списки, множества, массивы и таблицы. Попутно представим ряд команд для работы с данными этих типов.

Последовательность выражений — exprseq

Переменная типа `exprseq` получается как последовательность выражений Maple, разделенных запятыми. Например:

```
> ex:=2, 3, x^2, "abc", 2;
```

$ex := 2, 3, x^2, abc, 2$

Здесь и далее в тексте знаком «больше» (>), как уже говорилось, отмечено приглашение ввода. Порядок элементов для переменных типа exprseq сохраняется, так что могут встречаться одинаковые элементы. На основе последовательностей часто строятся новые объекты. Используя последовательности, можно организовывать множественные присваивания:

```
> a,b,c := 1, 12, 123;
```

$a, b, c := 1, 12, 123$

При помощи операции конкатенации (||) и переменной можно организовывать комбинированные имена. Организуем набор имен переменных, начинающихся с буквы А, используя диапазон и оператор конкатенации:

```
> LI:=1..3;
```

$LI := 1 .. 3$

```
> A||LI;
```

$A1, A2, A3$

Для объединения нескольких последовательностей выражений достаточно записать их через запятые:

```
> ab:= a, b, ex, "Sic";
```

$ab := 1, 12, 2, 3, x^2, abc, 2, Sic$

Последовательности выражений удобны для накопления уравнений, переменных и т. д. Для обозначения пустой последовательности имеется специальная константа NULL.

При помощи знака-повторителя $ можно создавать последовательности выражений из символов и чисел. Например:

```
> A$3, $11..13, d[k]$k=-1..1, k^3$k=1..3;
```

$A, A, A, 11, 12, 13, d_{-1}, d_0, d_1, 1, 8, 27$

Для организации последовательностей exprseq имеется команда seq(EX,K=N..M), первый параметр EX задает K-й член последовательности, а второй параметр определяет диапазон изменения целой переменной K. Приведем простой пример:

```
> seq(k,k=-1..11);
```

-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Команда seq позволяет также создавать последовательности выражений, состоящие из символьных величин. Например, можно сформировать систему уравнений с неизвестными d_1, d_2, d_3, d1, d2, d3 всего одной командой:

```
> eq:=seq(d[k]=k^2+d||k,k=1..3);
```

$eq := d_1 = 1 + d1, d_2 = 4 + d2, d_3 = 9 + d3$

В качестве второго параметра может выступать список (см. следующий раздел). Здесь это попросту последовательность выражений, взятая в квадратные скобки. Например, вычислить производную от ряда тригонометрических функций позволяет комбинация команды seq и оператора дифференцирования D (см. главу 3 «Математический анализ в Maple»):

```
> Df:=seq(D(f),f=[sin,cos,tan,cot,sec,csc]);
```

$Df := \cos, -\sin, 1 + \tan^2, -1 - \cot^2, \sec\tan, -\csc\cot$

Доступ к элементам последовательности выражений достаточно прост: для выбора элемента нужно указать его номер (целое положительное число), считая слева направо. При указании отрицательного числа нумерация идет от конца последовательности. Для выбора нескольких последовательных элементов нужно указать диапазон.

Выберем из переменной Df второй и предпоследний элементы, а также группу элементов от третьего до предпоследнего:

```
> Df[2]; Df[-2]; Df[3..-2];
```

$-\sin$

$\sec\tan$

$1 + \tan^2, -1 - \cot^2, \sec\tan$

Список — list

Последовательность выражений в квадратных скобках образует переменную типа list (список):

```
> ex:=2,3,x^2,'abc',2: lex:= [ex];
```

$lex := [2, 3, x^2, abc, 2]$

Тот же список получится по команде list(ex). Обращение к элементам типа list аналогично рассмотренному для последовательностей exprseq: в квадратных скобках указывается номер или диапазон номеров. Нумерация ведется слева направо, если номера положительные. Отрицательные числа применяются для указания порядкового номера справа налево.

Для работы со списками применяются команды выбора по заданному правилу (select) и удаления (remove). Приведем примеры выделения простых чисел (команда isprime) из списка цифр от двух до девяти:

```
> num:=[seq(k,k=2..9)];
```

$num := [2, 3, 4, 5, 6, 7, 8, 9]$

```
> LA:=select(isprime,num);
```

$LA := [2, 3, 5, 7]$

Применение команды позволяет получить список составных чисел:

```
> LB:=remove(isprime,num);
```

$LB := [4, 6, 8, 9]$

Для превращения списка в последовательность выражений достаточно поставить после имени переменной пару квадратных скобок:

```
> LA[];
```

$2, 3, 5, 7$

Тот же эффект получится при выполнении команды op(LA).

Объединить данные двух списков можно путем перехода к последовательностям при помощи команды

```
> LiAB:= [LA[],LB[]];
```

$LiAB := [2, 3, 5, 7, 4, 6, 8, 9]$

Того же результата можно добиться использованием команды op:

```
> LiAB:=[op(LA),op(LB)];
```

$LiAB := [2, 3, 5, 7, 4, 6, 8, 9]$

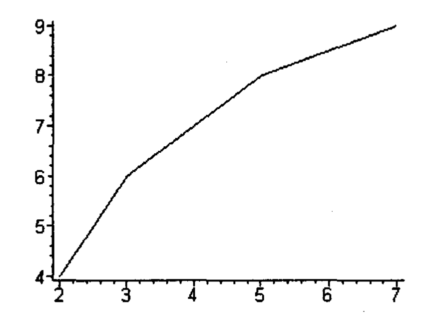
При помощи команды zip можно объединять списки по заданному правилу. Эта команда может быть использована для преобразования данных из разных списков в список пар (троек) для вывода графика. Например, образуем пары из элементов двух списков — A и B:

```
> AB:=zip((x,y)->[x,y],LA,LB);
```

$AB := [[2, 4], [3, 6], [5, 8], [7, 9]]$

В результате получается последовательность списков. Заключив переменную AB в квадратные скобки, получаем нужный для построения графика по точкам список списков [AB]. Сам график строится при помощи команды plot:

```
> plot([AB]);
```



Для сортировки элементов списка LIS используется команда

```
sort(LIS,FUN)
```

При помощи дополнительного параметра FUN — функции или термина — можно задать порядок следования элементов списка. Для числовых элементов термин "<" (действует по умолчанию) определяет упорядочивание в порядке возрастания, а термин ">" устанавливает обратный порядок. Для строковых и символьных элементов команда sort упорядочивает элементы списка в лексикографическом порядке (lexorder или string). Для термина address упорядочивание производится по адресам памяти, в которых расположены элементы списка.

Приведем примеры:

```
> sort([1,3,2]); sort([1,3,2],">");
```

[1, 2, 3]
[3, 2, 1]

```
> L:=["b1",b,"a2","b",a]: sort(L,string); sort(L,lexorder);
```

[*a*, "a2", *b*, "b", "b1"]
[*a*, "a2", *b*, "b", "b1"]

```
> L:=["b",1,a2,"a",2]: sort(L); sort(L,address);
```

[1, 2, "a", "b", *a2*]

[1, 2, "a", "b", *a2*]

Приведем пример сортировки с введением функции для указания правила, здесь используем команду оценки (вычисления) булевского выражения evalb, заданное правило позволяет упорядочить числа так, чтобы квадрат следующего числа превосходил значение текущего:

```
> sort([6,2,5,1],(x,y)->evalb(x<y^2));
```

L1 := [1, 2, 6, 5]

Множество — set

Заключив последовательность выражений в фигурные скобки, получим переменную типа set (множество). Именно в виде множеств удобно задавать системы уравнений и получать найденные Maple решения уравнений:

```
> ex:=2,3,x^2,'abc',2: sex:={ex};
```

sex := {2, 3, x^2, *abc*}

Уже из примера видно, что в объектах типа set удаляются одинаковые элементы. Множество есть неупорядоченная совокупность элементов. Команда set(EX) для последовательности EX эквивалентна действию фигурных скобок {EX}.

При работе с множествами необходимы операции объединения union, пересечения intersect, вычитания minus. Для образования пустого множества достаточно пары фигурных скобок:

```
> empt:={}; new:=empt union {abc,3};
```

empt := { }

new := {3, ***abc***}

```
> new intersect sex;
```

{3, ***abc***}

Имеются также полные аналоги этих операций — одноименные команды. В этом случае в качестве аргументов указываются имена множеств, участвующих в операции. Результат последней операции получится также после исполнения команды

```
> "intersect"(new,sex);
```

{3, ***abc***}

Массив — array

Массивы или объекты типа array позволяют организовывать данные, используя для индексации отрицательные числа и нуль. Массив создается по команде

```
array(FUN,DIA,LIS)
```

Параметры ее имеют следующее назначение: функция FUN задает свойства массива, переменная DIA определяет диапазоны изменения индексов, а LIS есть список элементов массива. Каждый из параметров может быть опущен, но по крайней мере один диапазон или список элементов должны быть заданы. В качестве

имени FUN могут быть использованы следующие стандартные методы: symmetric, antisymmetric, sparse, diagonal, identity, package. Это позволяет определить соответственно массивы симметричные и антисимметричные (кососимметричные), разреженные (нули для неупомянутых элементов), массивы с ненулевой диагональю и единичные. Имя package указывает на процедуру, служащую для ввода элементов.

Например, для создания массива из четырех элементов с именем A используется следующая конструкция:

```
> A:=array(-1..2);
```

$A := \mathrm{array}(-1 .. 2, [\ ])$

Сама переменная A при этом считается строковой (string), а любой элемент массива — индексной переменной (indexed). С элементами можно работать, как с обычными переменными: присваивать им значения, использовать при организации выражений. Например:

```
> A[-1]:=qwerty: A[1]:='йцукен': A[0]:=0: A[1]:=A[0]-A[2];
```

$A_1 := -A_2$

Чтобы посмотреть содержимое массива, используем команду print:

```
> print(A);
```

array(-1 .. 2, [
    $(-1) = qwerty$
    $(0) = 0$
    $(1) = -A_2$
    $(2) = A_2$
    ])

Отметим, что один из элементов массива так и остался с начальным содержимым — с именем A_2.

Кстати, использование переменной с индексом вовсе не означает, что это элемент массива. Можно вводить переменные с индексом без предварительного описания массива. Пример такого рода дан в разделе, посвященном последовательностям выражений.

Таблица — table

При помощи команды table можно организовать данные в виде таблицы, отказавшись от целочисленной нумерации:

```
table(FUN,LIS)
```

Здесь функция FUN определяет свойства таблицы, а список элементов LIS формируется в виде пар равенств: Индекс=Значение. Для задания свойства можно использовать методы, перечисленные ранее.

Например, создадим таблицу, устанавливающую соответствие между цифрами в английском и итальянском языках:

```
> T:=table(sparse,[one=uno,two=due,three=tre]);
```

T := table(*sparse*, [*three* = *tre*, *one* = *uno*, *two* = *due*])

Метод заполнения таблицы sparse выбран для того, чтобы при обращении с именем, которого нет в таблице, выдавался бы нуль. Действительно:

```
> T[one]; T[five];
```

uno

0

Если же сформировать таблицу без указания метода, то при аналогичном обращении получим:

```
> Ta:=table([one=uno,two=due,three=tre]); Ta[five];
```

Ta := table([*three* = *tre*, *one* = *uno*, *two* = *due*])

Ta_{five}

Элементами таблицы, в свою очередь, могут быть таблицы, а также множества, списки и массивы. Таким способом обеспечивается возможность определения разнородных данных, например, таблиц физических констант, записанных в разных системах единиц.

Сложные типы данных

Комбинируя данные разных типов, можно получить, например, списки множеств или последовательности списков и т. д. К составным типам относятся indexable, sequential, tabular, rtable. Встроенными типами (built-in) являются And, Or, Not, nonnegint, posint, nonposint, negint, negative, positive, nonnegative, nonpositive. Большое число типов связано с необходимостью адекватно обрабатывать многообразие математических конструкций. Кроме того, в дополнение к имеющимся типам можно создавать свои, расширяя возможности Maple для решения возникающих задач.

Для работы с данными и извлечения элементов полезны команды nops (число операндов) и op (взятие операнда), см. главу 2 «Аналитические преобразования в Maple».

Команда map и простые команды работы со списками

Опишем действие ряда команд, позволяющих эффективно обрабатывать данные (числа, выражения и другие), составляющие списки. Очень важной является команда

```
map(FUN,LIS)
```

Эта команда применяет функцию FUN к каждому элементу объекта LIS, в качестве которого может выступать список, множество или выражение. Вычисление косинуса для нескольких значений из списка демонстрирует следующий пример:

```
> map(cos,[0,Pi/2,Pi]);
```

[1, 0, -1]

Кроме того, применяемая функция может быть задана непосредственно в теле команды map. Например:

```
> map(x->x^3,[-1,2,a]);
```

$[-1, 8, a^3]$

При помощи команды map можно формировать объекты с дополнительными элементами. Это достигается введением третьего и последующих параметров. Например, можно организовать последовательность обращений к функции f с тремя параметрами, где два аргумента одинаковы, а меняется только первый аргумент:

```
> map(f,[a,b],0,d);
```

$[f(a, 0, d), f(b, 0, d)]$

Чтобы вставить дополнительный параметр впереди, имеется команда map2, работу которой демонстрирует следующий пример:

```
> map2(f,A,[a,b],d);
```

$[f(A, a, d), f(A, b, d)]$

Для суммирования элементов имеется команда add, а для вычисления произведения — mul. Приведем пример:

```
> add(f,f=[0,Pi,I]); mul(f,f=[1,Pi,I]);
```

$I + \pi$

$I\pi$

Несколько простых команд общего назначения не требуют особого комментария. Для вычисления длины объекта OBJ (строки, списка, выражения) используется команда

```
length(OBJ)
```

Для целого числа будет выведено число цифр, для строки — число символов, для rtable — число слов, а для переменных других типов вычисление будет проведено рекурсивно и ответ будет некоторой характеристикой участвующих в данном объекте базисных элементов.

Приведем примеры:

```
> length("Hello");
```

5

```
> length([1,2]);
```

7

```
> length(a+b+c);
```

13

Для нахождения максимума и минимума из выражений EX1,EX2,... применяются соответственно команды

```
max(EX1,EX2,...)
```

и

```
min(EX1,EX2,...)
```

Приведем пример:

```
> m:=max(1.5,min(2,3),a);
```

$m := \max(2, a)$

Поскольку в перечне величин присутствовала одна переменная, то ответ Maple состоит из максимального значения для перечисленных чисел и этой переменной. Присвоив значение переменной a, выведем значение максимума:

```
> a:=5: "max"=m;
```

$max = 5$

Стандартные математические функции

Обращение к стандартной функции — элементарная операция, если известно имя нужной функции. Список изначально определенных в Maple функций велик, поэтому просто перечислим достаточно представительное подмножество из известных функций. В табл. 1.11 и 1.12 указаны основные имена функций Maple и соответствующие математические функции.

Таблица 1.11. Стандартные функции

Maple	Математическая запись
exp(x)	e^x
ln(x) или log(x)	$\ln x$
log10(x)	$\log_{10} x$
log[a](x)	$\log_a x$
sqrt(x)	$\sqrt{x}$
abs(x)	$\|x\|$
signum(x)	$\operatorname{sgn} x$
n!	$n!$

Например:

```
> exp(1); ln(exp(3));
```

e

3

Для преобразования вещественных чисел в целые имеется необходимый набор команд, см. табл. 1.12. Для комплексных величин команда применяется отдельно к реальной и мнимой части.

Таблица 1.12. Функции округления

Имя	Назначение
round	Округление к ближайшему целому
trunc	Округление отбрасыванием дробной части
floor	Округление к меньшему целому
ceil	Округление к большему целому

Приведем примеры работы этих функций, для компактности вывода организовав последовательность выражений из результатов вычислений:

```
> x:=-1.4: round(x),trunc(x),floor(x),ceil(x);
```

-1, -1, -2, -1

```
> x:=-1.5: round(x),trunc(x),floor(x),ceil(x);
```

-2, -1, -2, -1

```
> x:=1.4: round(x),trunc(x),floor(x),ceil(x);
```

1, 1, 1, 2

```
> x:=1.5: round(x),trunc(x),floor(x),ceil(x);
```

2, 1, 1, 2

Для вычисления дробной части выражения EX имеется команда frac(EX). Проиллюстрируем ее возможности на примере вычисления комплексных выражений:

```
> frac(2.001-I*1.1);
```

$.001 - .1\, I$

```
> frac(sqrt(2)+I*Pi);
```

$\sqrt{2} + I\pi - 1 - 3\, I$

Обратим внимание на квалификацию Maple при выполнении последней команды: верный ответ записан с использованием иррациональных чисел.

В табл. 1.13 перечислим семейства функций, для которых достаточно указать имя. Для ряда функций помимо скалярного аргумента (x) должен быть указан порядок (p). Справку о всех имеющихся в Maple функциях можно получить, выполнив команду

```
?inifunction
```

Таблица 1.13. Математические функции

Имена	Назначение
sin, cos, tan, cot, sec, csc	Тригонометрические функции (аргументы в радианах)
arcsin, arccos, arctan, arccot, arcscs, arccsc	Обратные тригонометрические функции
sinh, cosh, tanh, coth, sech, csch	Гиперболические функции
arcsinh, arccosh, arctanh, arccoth, arcsech, arccsch	Обратные гиперболические функции
Dirac	Дельта-функция Дирака
Heaviside	Функция Хевисайда
BesselJ(p,x), BesselY(p,x), BesselI(p,x), BesselK(p,x)	Бесселевы функции
GAMMA(z), GAMMA(a,x), Beta(x,y)	Гамма- и бета-функции
Zeta(z)	Дзета-функция Римана
erf(x)	Интеграл ошибок

продолжение ➭

Таблица 1.13 (*продолжение*)

Имена	Назначение
LegendreF, LegendreE, LegendrePi	Эллиптические интегралы в форме Лежандра первого, второго и третьего рода
LegendreKc, LegendreEc, LegendrePic	Полные эллиптические интегралы в форме Лежандра первого, второго и третьего рода
round, trunc	Округление и усечение к целому

Точные и приближенные вычисления

По умолчанию Maple проводит вычисления с целыми и рациональными числами, радикалами и константами, не переходя к машинной арифметике, что позволяет проводить их точно. Например:

```
> a:=sqrt(2): b:=sin(Pi/17):
> c:=a^13+b/a;
```

$$c := 64\sqrt{2} + \frac{1}{2}\sin\left(\frac{1}{17}\pi\right)\sqrt{2}$$

Maple автоматически переходит к операциям с плавающей запятой в том случае, если в выражении присутствуют числа, определенные в формате с десятичной точкой, но в этом случае вычисления выполняются с погрешностью округлений. Пример:

```
> 26^(1.0/6);
```

1.721190306

```
> %^6;
```

26.00000003

В общем случае для перехода к арифметике с плавающей запятой необходимо дать соответствующую команду. Одной из таких команд является `evalf`, которая в качестве первого параметра имеет вычисляемое выражение, а необязательный второй параметр определяет мантиссу, с которой будут проводиться эти вычисления. По умолчанию мантисса определяется системной константой `Digits`. Вычислим значение переменной с мантиссой 50 знаков:

```
> evalf(c,50);
```

90.639598521965938336586291285766156688985958173372

Используя команду `evalf` и переопределяя значение константы `Digits`, можно проводить операции с плавающей запятой практически с любой мантиссой. Однако следует помнить, что увеличение мантиссы сильно замедляет вычисления. Для быстрых вычислений с плавающей запятой имеется команда `evalhf`. Эта команда использует арифметику процессора напрямую, но все операции происходят с машинной мантиссой используемого компьютера. Проиллюстрируем время выполнения операций для различных вариантов арифметики с плавающей запятой. Для оценки времени использована команда `time`, которая будет описана ниже. Вычис-

лим тысячу раз функцию sin для различных аргументов с использованием команд `evalf` и `evalhf`:

```
> time([seq(evalf(sin(i)),i=1..1000)]);
```

1.445

```
> time([seq(evalhf(sin(i)),i=1..1000)]);
```

.044

Теперь переопределим константу `Digits` и снова обратимся к `evalf`:

```
> Digits:=15:
> time([seq(evalf(sin(i)),i=1..1000)]);
```

2.702



Аналитические преобразования в Maple

Наиболее часто используемые основные математические и общесистемные команды находятся в ядре Maple и стандартной библиотеке, которая автоматически подключается после запуска программы. Предваряя описание команд для выполнения основных математических операций (интегрирования, дифференцирования, разложения в ряды, суммирования, решения уравнений и неравенств и т. д.), остановимся на командах общего назначения, позволяющих приводить выражения к нужному виду, упрощать их, делать подстановки, и т. д.

Сразу отметим, что ряд преобразований, основанных на свойствах тригонометрических, логарифмических и других стандартных функций, Maple выполняет непосредственно, без дополнительных указаний. Продемонстрируем это на элементарном примере формирования уравнения:

```
> eq:=sin(Pi+x)=x/ln(exp(-Pi));
```

$$eq := -\sin(x) = -\frac{x}{\pi}$$

При записи уравнения сразу были использованы формула приведения для синуса и обратимость логарифмической и показательной функций. Вместе с тем в левой и правой частях уравнения остались минусы, и для упрощения полезен еще один шаг:

```
> eq:=-eq;
```

$$eq := \sin(x) = \frac{x}{\pi}$$

Для проведения не столь тривиальных преобразований требуются дополнительные усилия и нужно иметь представление об организации записи выражений (структурах объектов), знать основные команды преобразования формул и их возможности. Вначале коротко рассмотрим внутреннюю организацию выражений в Maple, преобразование объектов от одного типа к другому, семейство команд оценивания и вычисления выражений. Затем опишем команды для операций с формулами (упрощение, приведение к нужному виду и другие) и рассмотрим команды для работы с полиномами.

Структура выражений

Во внутреннем представлении Maple каждый объект (формула, уравнение, таблица и др.) состоит из подобъектов, каждый из которых, в свою очередь, может состоять из подобъектов, и т. д., вплоть до базисных элементов, так что получается ветвящаяся, древовидная структура. При работе с большими выражениями часто требуется извлекать отдельные элементы структуры и преобразовывать их. Команда nops выводит число элементов первого уровня, а команда op выводит их в виде последовательности выражений.

Проиллюстрируем сказанное простым примером:

```
> ob:=x^3-y+sin(z^2);
```

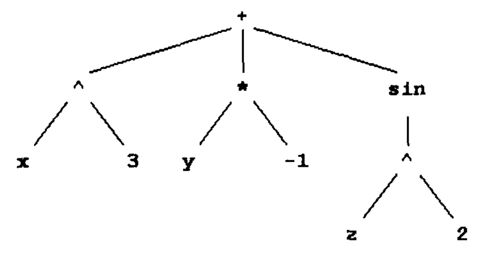
$ob := x^3 - y + \sin(z^2)$

```
> nops(ob);op(ob);
```

3

$x^3, -y, \sin(z^2)$

Объект Maple хранится в виде древовидной структуры, в узлах которой находятся операции (+, *, ^, sin), а ветви указывают на операнды. Для введенного выражения ob дерево схематически выглядит следующим образом:



При помощи команды op можно извлекать подвыражения, указывая номер операнда верхнего уровня. Для отсчета подвыражений справа налево используются отрицательные числа. Например:

```
> op(3,ob),op(2..-1,ob);
```

$\sin(z^2), -y, \sin(z^2)$

Для доступа к операнду операнда в качестве первого параметра команды op следует определить список, где числа слева направо обозначают номера операндов, начиная от верхнего уровня. Для вывода второго подоперанда второго операнда выражения ob достаточно задать

```
> op([2,2],ob);
```

y

Для подстановки выражения NEW в n-й операнд первого уровня выражения EX используется команда

```
subsop (n=NEW,EX)
```

Представим простую иллюстрацию действия данной команды для выражения ob:

```
> subsop(3=Pi,ob);
```

$x^3 - y + \pi$

Типы и их преобразование

Может получиться, что Maple выведет выражение не в той форме, которую вы ожидали или предпочитаете. Чтобы упростить выражение, перейти от экспоненциальных функций к тригонометрическим и для многих других преобразований, применяется команда convert. Кроме того, каждая команда Maple работает с данными определенных типов, поэтому если тип фактического параметра не отвечает типу, допустимому для данной команды, то будет выведено сообщение об ошибке. В этом случае перед использованием команды следует преобразовать выражение или объект, приведя его к нужному типу. Определить тип выражения EX помогает команда

```
whattype (EX)
```

Результатом ее выполнения будет вывод одного из терминов, перечень которых дан в табл. 2.1. Чтобы выяснить, является ли выражение EX объектом типа kind, используется команда

```
type (EX,kind)
```

В качестве kind может выступать один из терминов, приведенных в табл. 2.1, а также многие другие типы, полную и адекватную информацию о которых предоставит справка Maple. Квалифицированный пользователь может определить также и свои типы.

Более полную информацию о термине kind можно получить, обратившись к справке Maple:

```
?type[kind]
```

Приведем ряд примеров, иллюстрирующих действие команд whattype и type:

```
> whattype(1.2e3),whattype(1/2),whattype(sin(x));
```

float, fraction, function

```
> f:=x+y: whattype(f),whattype("f"),whattype(''f'');
```

+, *symbol, uneval*

```
> whattype(Pi>3),whattype(0..1),whattype(a[k]);
```

<, .., *indexed*

```
> whattype(a,b),whattype([a,b]),whattype({a,b});
```

exprseq, list, set

```
> whattype(<1,2>),whattype(<1|2>),whattype(<<1,2>,<3,4>>);
```

$Vector_{column}$, $Vector_{row}$, *Matrix*

```
> type(a-b,"+"),type(a/b,"*"),type(a and b,"and");
```

true, true, true

```
> type(3,positive),type(sin(x*y),trig),type(sin(x)+1,trig);
```

true, true, false

При анализе выражений полезна команда has(EX,VAR), позволяющая определить, содержит ли объект EX выражение или переменную VAR. Результатом выполнения команды является булево выражение. Проверить, входит ли в объект EX объект типа TYP, позволяет команда hastype(EX,TYP). Эти команды особенно полезны при программировании, когда для работы существенна информация о типе или со-

ставных частях выражения (см. вывод формулы метода Рунге–Кутты в главе 10 «Примеры решения задач»). Приведем простую иллюстрацию действия этих команд:

```
> g:=1-x^2: has(g,x^2); hastype(g,function);
```

true

false

Таблица 2.1. Перечень типов

Термин	**Значение**
`"*"`	Произведение выражений
`"+"`	Сумма выражений
`"^"`	Степень
`".."`	Диапазон
`"::"`	Определение типа
`"<","<=", "<>","="`	Операция отношения
`"and", "not", "or"`	Булевы операции
`array`	Массив
`exprseq`	Последовательность выражений
`float`	Вещественное число
`fraction`	Дробь
`function`	Функция
`hfarray`	Массив для вычислений, использующих арифметический процессор компьютера
`indexed`	Индексная величина
`integer`	Целое число
`list`	Список
`procedure`	Процедура
`series`	Разложение
`set`	Множество
`string`	Строка
`symbol`	Символьная величина
`table`	Таблица
`uneval`	Невычисленное (неоцененное) выражение
`Array`	Массив для команд из пакета `LinearAlgebra`
`Matrix`	Матрица для команд из пакета `LinearAlgebra`
`Vector[column]`, `Vector[row]`	Вектор-столбец и вектор-строка для команд из пакета `LinearAlgebra`

Теперь перейдем к команде `convert`, назначение которой — преобразовывать одни объекты в другие с изменением их типа. Для преобразования выражения `EX` согласно заданному параметру `OPT` применяется вызов

```
convert (EX,OPT)
```

В качестве параметра OPT могут выступать следующие термины: "+", "*", D, base, binary, binomial, confrac, decimal, degrees, diff, double, equality, exp, expln, expsincos, factorial, float, fraction, GAMMA, hex, horner, hostfile, hypergeom, lessthan, lessequal, list, listlist, ln, matrix, metric, mod2, multiset, name, octal, parfrac, polar, polynom, radians, radical, rational, ratpoly, RootOf, set, sincos, sqrfree, tan, trig, vector.

Отметим, что назначение ряда терминов разъяснено ранее или будет указано далее, мнемоника помогает понять смысл других терминов (binary, diff, radians). За получением точного значения любого термина termin следует обратиться к справочной системе, набрав команду

```
?convert[termin]
```

Приведем характерные случаи употребления команды convert:

- convert(LIS,array) — преобразование списка LIS в одномерный массив;
- convert(LIS_1,...,LIS_N,matrix) — преобразование списков LIS_1, ..., LIS_N в матрицу;
- convert(LIS,ratpoly) — преобразование списка LIS в полиномиальное выражение с рациональными коэффициентами;
- convert(P,list) — преобразование полинома P в список.

Кроме того, о некоторых разновидностях команды convert будет рассказано ниже.

Следующий пример показывает, как, вычислив разложение функции cos в ряд Тейлора (см. главу 3 «Математический анализ в Maple»), перевести его в полином, воспользовавшись для этого командой convert:

```
> s:=series(cos(x),x,7);
```

$$s := 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + O(x^7)$$

```
> p:=convert(s,polynom);
```

$$p := 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6$$

После преобразования списка (list) в переменную типа множество (set) число элементов может уменьшиться, так как Maple удаляет из множеств одинаковые элементы. Это можно использовать для формирования списка без одинаковых элементов:

```
> S1:=convert([2,0,0,1],set); convert(S1,list);
```

$$S1 := \{0, 1, 2\}$$

$$[0, 1, 2]$$

Для перевода тригонометрического выражения к экспоненциальной форме и обратно также можно использовать команду convert:

```
> f:=exp(x+I*Pi)+exp(I*x); convert(f,trig); convert(%,exp);
```

$$f := e^{(x + I\pi)} + e^{(I x)}$$

$$-\cosh(x) - \sinh(x) + \cos(x) + I\sin(x)$$

$$-e^x + e^{(I x)}$$

Вычисление выражений

Целое семейство команд посвящено операциям оценивания или вычисления выражений: eval, evalf, evalm, evalc и др.

Чтобы получить текущее значение переменной скалярного типа, достаточно в строке ввода указать ее имя. Для просмотра содержимого таких структур данных, как таблицы, матрицы и процедуры, нужно использовать команду печати print или команду eval.

Поясним это на примере. Определим массив A:

```
> A:=array(1..2,1..2,[[a,b],[c,a]]):
```

Если в строке ввода указать просто имя A, то вывода содержимого массива не последует. Для вывода применим команду eval:

```
> A; eval(A);
```

A

$$\begin{bmatrix} a & b \\ c & a \end{bmatrix}$$

Поскольку индексы массива A принимают положительные значения, то результат выведен в виде матрицы. Следовательно, вывести содержимое A можно было и при помощи команды оценки матриц evalm.

Команда eval необходима для подстановки значений в переменные, участвующие в определении массива:

```
> subs(c=13,eval(A));
```

$$\begin{bmatrix} a & b \\ 13 & a \end{bmatrix}$$

Для массивов с неположительными значениями индексов вывод содержимого производится в табличной форме. Например:

```
> B:=array(-1..0,[a,b]): subs(b=sin(Pi+d),eval(B));
```

array(-1 .. 0, [
    (-1) = a
    (0) = $-\sin(d)$
    ])

Этот пример напоминает также, что в Maple упрощение выражений с функциями производится и при подстановках (использована формула приведения для синуса).

Многоцелевая команда eval(EX,EQ) позволяет оценить значение функции EX при значениях переменных, определенных уравнением (множеством уравнений), без присваивания. Например:

```
> eval(sin(z)+3*z+y,{z=Pi/3,y=a}); y,z;
```

$$\frac{1}{2}\sqrt{3}+\pi+a$$

y, z

По умолчанию в Maple проводится полное оценивание для глобальных переменных и выполняется одношаговая оценка для локальных переменных процедур, см. главу 7 «Программирование в Maple».

При работе в Maple нужно иметь в виду, что система запоминает назначения. Например:

```
> x:=y: y:=z: z:=7: x,y,z;
```

7, 7, 7

Для выяснения вложенности назначений переменной применяется команда eval с целым числом в качестве второго параметра для указания уровня вложенности. Так, для определенной выше переменной x имеем:

```
> seq(eval(x,k),k=1..3);
```

$y, z, 7$

Если переопределить одну из участвующих в определении x переменных, то получим:

```
> y:=Pi: x,y,z;
```

$\pi, \pi, 7$

Ряд команд семейства eval служит для вычисления выражения с преобразованием к нужному типу. Если в выражении EX все числа заданы при помощи рациональных величин (дроби, степени), то для перехода к числам с плавающей запятой применяется команда

```
evalf (EX)
```

Кроме того, для ускорения вычислений с плавающей запятой имеется команда evalhf, использующая арифметику компьютера, то есть при этом все вычисления происходят на аппаратном уровне Например, использование арифметического процессора позволяет на порядок сократить время вычисления последовательности значений синуса:

```
> time( seq( sin(k*0.1) ,k=1..10^4) );
```

13.330

```
> time( seq( evalhf(sin(k*0.1)) ,k=1..10^4) );
```

1.170

Здесь использована команда time, позволяющая оценить (в секундах) время, затраченное на вычисление заданного выражения или команды.

Для вычисления матричного выражения EX с матрицами в качестве операндов и допустимыми операторами &*, +, *, ^ применяется команда

```
evalm(EX)
```

Для вычисления значения комплексного выражения EX применяется команда

```
evalc (EX)
```

Операции с комплексными величинами

В Maple комплексные числа представляются собственными структурами данных, благодаря чему быстрее выполняются операции комплексной арифметики. Для

работы с комплексными числами необходимы функции (команды), перечисленные в табл. 2.2 (CX обозначает комплексное выражение).

Таблица 2.2. Функции для работы с комплексными числами

Имя	Назначение
conjugate(CX)	Вычисление сопряженной величины для комплексного выражения
Im(CX)	Определение мнимой части выражения
polar(CX)	Преобразование комплексного выражения в полярные координаты
Re(CX)	Определение действительной части выражения

Приведем примеры, использующие некоторые из этих функций:

```
> c:=2*(x+I*y)^2/5+sin(Pi*sqrt(2001)); evalf(c);
```

$$c := \frac{2}{5}(x + I\,y)^2 + \sin(\pi\sqrt{2001})$$

$$.4000000000\,(x + 1.\,I\,y)^2 + .7448133732$$

```
> Re(c); evalc(Re(c));
```

$$\sin(\pi\sqrt{2001}) + \frac{2}{5}\Re((x + I\,y)^2)$$

$$\sin(\pi\sqrt{2001}) + \frac{2}{5}x^2 - \frac{2}{5}y^2$$

Операции с формулами

Преобразуя математические выражения, обычно приходится делать множество рутинных операций: приводить подобные члены, раскладывать на множители (факторизовывать), раскрывать скобки, делать подстановки и т. д. Для проведения выкладок Maple обладает широкими возможностями, и сложные аналитические преобразования математических формул выполняются при помощи ряда весьма эффективных команд. Имена команд для проведения аналитических преобразований просты и соответствуют английским терминам: факторизация — factor, раскрытие скобок — expand, упрощение выражения — simplify, подстановка — subs и т. д.

Проиллюстрируем действие некоторых команд работы с алгебраическими выражениями. Присвоим переменной w дробь, числитель и знаменатель которой являются алгебраическими выражениями от x и y:

```
> w:=(x^4-y^4)/(x^6+y^6);
```

$$w := \frac{x^4 - y^4}{x^6 + y^6}$$

Разложим на множители это выражение, используя команду factor:

```
> z:=factor(w);
```

$$z := \frac{(x - y)(y + x)}{x^4 - x^2 y^2 + y^4}$$

Здесь помимо факторизации выполнено сокращение одинаковых множителей дроби. Для сокращения (нормализации) дроби используется команда normal, а команды numer и denom позволяют выделить соответственно числитель и знаменатель дроби:

```
> numer(z); d:=denom(z);
```

$$(x-y)(y+x)$$

$$d := x^4 - x^2 y^2 + y^4$$

Далее проведем подстановку в выражении d, заменив x и y тригонометрическими функциями аргумента t:

```
> u:=subs({x=cos(t),y=sin(t)},d);
```

$$u := \cos(t)^4 - \cos(t)^2 \sin(t)^2 + \sin(t)^4$$

При помощи команды simplify упростим полученное выражение:

```
> simplify(u);
```

$$3\cos(t)^4 - 3\cos(t)^2 + 1$$

Наконец, используя команду combine, преобразуем выражение u к более компактному виду:

```
> combine(u);
```

$$\frac{3}{8}\cos(4\,t) + \frac{5}{8}$$

Как видно из примера, в Maple принят удобный формат команд и проведение выкладок достаточно просто и естественно.

В табл. 2.3 в алфавитном порядке перечислены основные команды, применяемые для упрощения выражений, и использованы следующие обозначения: EX — выражение, VAR — переменная или список переменных, RAT — рациональная дробь, OPT — дополнительные условия.

Таблица 2.3. Команды упрощения

Обращение	Назначение
collect(EX,VAR)	Приведение подобных членов
combine(EX,VAR)	Объединение членов
expand(EX)	Раскрытие скобок
factor(EX)	Разложение выражения на множители
normal(RAT)	Нормализация дроби
simplify(EX,OPT)	Упрощение выражения

Приведем пояснения для перечисленных в табл. 2.3 команд, сопроводив изложение характерными примерами.

Раскрытие скобок — expand

Для полиномиальных выражений EX команда expand(EX) раскрывает скобки, а дробно-рациональные выражения приводит к виду, пригодному для последующего интегрирования. Например:

```
> expand(y*(y+1)+(x+2)/(x+1));
```

$$y^2+y+\frac{x}{x+1}+\frac{2}{x+1}$$

Команда expand использует информацию о многих математических функциях, что позволяет применять ее для упрощения тригонометрических, логарифмических и показательных функций:

```
> expand(tan(a+b)+exp(ln(Z)*D));
```

$$\frac{\tan(b)+\tan(a)}{1-\tan(b)\tan(a)}+Z^D$$

Приведение членов — collect

Для приведения подобных членов (группировки по степеням) в выражении EX относительно переменной VAR используется команда collect(EX,VAR). В качестве переменных, при которых группируются коэффициенты, может фигурировать список, а также имена тригонометрических и иных функций, операция дифференцирования diff и т. д. Maple рассматривает выражение EX как полином относительно переменных, определенных параметром VAR, и собирает коэффициенты при одинаковых рациональных (положительных, отрицательных и дробных) степенях.

Например, определим выражение от двух переменных x и y:

```
> restart: f:=(x^(1/2)-y)*(y/x-1);
```

$$f:=(\sqrt{x}-y)\left(\frac{y}{x}-1\right)$$

Используем команду collect, чтобы записать тождество, в левой части которого приведение подобных членов дано по степеням x, а в правой — по степеням y:

```
> collect(f,x)=collect(f,y);
```

$$-\sqrt{x}+y+\frac{y}{\sqrt{x}}-\frac{y^2}{x}=-\frac{y^2}{x}+\left(\frac{1}{\sqrt{x}}+1\right)y-\sqrt{x}$$

Покажем, как преобразуется выражение при указании функции, при степенях которой следует собрать коэффициенты:

```
> ps:=expand(sin(x)+cos(x-y)); collect(ps,sin(x));
```

$$ps:=\sin(x)+\cos(x)\cos(y)+\sin(x)\sin(y)$$

$$(1+\sin(y))\sin(x)+\cos(x)\cos(y)$$

Теперь продемонстрируем использование дополнительных параметров. Определим выражение от двух переменных и соберем коэффициенты при переменной x:

```
> p:=x*y+x^2+y^2*x+(x*y)^2-y*x^2;
```

$$p:=x\,y+x^2+y^2\,x+x^2\,y^2-y\,x^2$$

```
> collect(p,x);
```

$$(1+y^2-y)\,x^2+(y+y^2)\,x$$

Соберем выражения при переменной y; параметр factor указывает на необходимость представления коэффициентов в виде произведения:

```
> collect(p,y,factor);
```

$$x(1+x)y^2 - x(-1+x)y + x^2$$

Покажем, как преобразуется выражение при указании нужного порядка следования переменных:

```
> collect(p,[y,x],recursive);
```

$$(x+x^2)y^2 + (x-x^2)y + x^2$$

В общем случае команда collect не проводит сортировки членов. Сопутствующими для collect являются команды вычисления коэффициентов coeff и coeffs, которые описаны в разделе, посвященном работе с полиномами.

Разложение на множители — factor

Разложение выражения EX на множители проводит команда factor(EX). По умолчанию коэффициенты разложения определяются тем полем, какому принадлежат коэффициенты исходного выражения. При помощи дополнительного параметра можно указать, чтобы в разложении фигурировали вещественные (real) и комплексные числа (complex), а также дать список радикалов, которые можно использовать при записи ответа.

Введем полином с целочисленными коэффициентами и единственным параметром a. Присвоим параметру целое значение и обратимся к команде factor:

```
> f:=x^6-4*x^4+x^2+a; a:=6; factor(f);
```

$$f := x^6 - 4x^4 + x^2 + a$$

$$(x^2-3)(x^2-2)(x^2+1)$$

Снова обратимся к команде factor, указав список допустимых при записи ответа радикалов:

```
> factor(f,[sqrt(3),sqrt(5)]);
```

$$(x^2+1)(x^2-2)(x+\sqrt{3})(x-\sqrt{3})$$

Если разрешить использование комплексных чисел, то будет получено следующее разложение (используем для компактности результата уменьшение числа значащих цифр посредством переопределения константы Digits):

```
> Digits:=4; factor(f,complex);
```

$$Digits := 4$$

$$(x+1.732)(x+1.414)(x+1.\,I)(x-1.\,I)(x-1.414)(x-1.732)$$

Если присвоить параметру a вещественное значение, то получим:

```
> a:=6.0: factor(f);
```

$$(x+1.732)(x+1.414)(x-1.414)(x-1.732)(x^2+1.)$$

Команда factor обеспечивает также сокращение подобных членов в алгебраической дроби:

```
> x9:=(x^9-1)/(x-1): x9=factor(x9);
```

$$\frac{x^9-1}{x-1}=(x+x^2+1)(x^6+x^3+1)$$

Нормализация дроби — normal

Для сокращения общих множителей дроби RAT и приведения выражения к общему знаменателю служит команда нормализации дроби normal(RAT). Для дроби, рассмотренной в предыдущем примере, команда normal дает менее компактный результат:

```
> x9=normal(x9);
```

$$\frac{x^9-1}{x-1}=x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$$

Это показывает, что для успешной работы нужно пользоваться разными командами упрощения. Команда normal действует рекуррентно для функций, множеств и списков. Например:

```
> normal([(x^2-1)*cos(y/(y+1)-y)/(1-x),y^(-2)-1/y+y]);
```

$$\left[-\cos\left(\frac{y^2}{y+1}\right)(x+1), \frac{1-y+y^3}{y^2}\right]$$

При помощи дополнительного параметра (expanded) можно указать, что для числителя и знаменателя должны быть раскрыты скобки:

```
> f:=g(x)/g(x)^2+1/(g(x)+1): normal(f)=normal(f,expanded);
```

$$\frac{2\,\mathrm{g}(x)+1}{\mathrm{g}(x)\,(\mathrm{g}(x)+1)}=\frac{2\,\mathrm{g}(x)+1}{\mathrm{g}(x)^2+\mathrm{g}(x)}$$

Кроме того, для удаления дробных степеней в знаменателе имеется команда rationalize:

```
> rationalize(1/(x^(5/3)+x^(-2/3)));
```

$$\frac{x^{(2/3)}\,(x^{(14/3)}-x^{(7/3)}+1)}{x^7+1}$$

Объединение выражений — combine

Команда combine(EX,FUN) преобразует выражение EX, стремясь к компактности результата, для чего использует правила упрощения многих математических функций. В некотором смысле команда combine противоположна команде expand. Дополнительный параметр FUN позволяет явно определить тип объекта, для которого будут проводиться преобразования. Для выражений, содержащих суммы, интегралы и пределы, происходит объединение под знаками суммы, интеграла и, соответственно, предела. Для тригонометрических функций используются формулы, объединяющие несколько функций в одну. Например:

```
> p:=Int(cos(x)*cos(2*b),x)-Int(sin(x)*sin(2*b),x);
```

$$p := \int \cos(x)\cos(2\,b)\,dx - \int \sin(x)\sin(2\,b)\,dx$$

```
> combine(p);
```

$$\int \cos(x + 2\,b)\,dx$$

В качестве параметра FUN может выступать один из следующих терминов: abs, arctan, conjugate, exp, icombine, ln, piecewise, polylog, power, Psi, radical, range, signum, trig. В этом случае для преобразования будут использоваться правила только для функций, специфицированных данным термином.

Например, запишем выражение с радикалами:

```
> f1:=sqrt(4-sqrt(3))*sqrt(4+sqrt(3));
```

$$f1 := \sqrt{4-\sqrt{3}}\,\sqrt{4+\sqrt{3}}$$

Обращение к combine позволяет собрать сомножители под знаком корня:

```
> f2:=combine(f1,radical);
```

$$f2 := \sqrt{(4-\sqrt{3})\,(4+\sqrt{3})}$$

После этого легко получить компактный ответ:

```
> factor(f2),expand(f2);
```

$$\sqrt{13}, \sqrt{13}$$

В то же время попытка непосредственного применения команд factor и expand к исходному выражению не приводит к полученному результату:

```
> factor(f1),expand(f1);
```

$$\sqrt{4-\sqrt{3}}\,\sqrt{4+\sqrt{3}}, \sqrt{4-\sqrt{3}}\,\sqrt{4+\sqrt{3}}$$

Для алгебраических выражений часто помогает использование в качестве дополнительного параметра термина symbolic. В этом случае при преобразовании не будут учитываться ограничения, накладываемые функцией на свои аргументы. Приведем пример, в котором под знаком корня стоят алгебраические выражения:

```
> f1:=sqrt(a-sqrt(a^2-b))*sqrt(a+sqrt(a^2-b));
```

$$f1 := \sqrt{a-\sqrt{a^2-b}}\,\sqrt{a+\sqrt{a^2-b}}$$

```
> f2:=combine(f1,radical,symbolic);
```

$$f2 := \sqrt{(a-\sqrt{a^2-b})\,(a+\sqrt{a^2-b})}$$

```
> expand(f2);
```

$$\sqrt{b}$$

Выделение частей выражения

В процессе преобразований может потребоваться выделить левую или правую часть уравнения, определить числитель или знаменатель дроби, выразить некоторое выражение из уравнения. Для этих действий применяются команды, перечисленные в табл. 2.4, где приняты следующие обозначения: EX — выражение, EQN — уравнение или диапазон, RAT — рациональная дробь.

Таблица 2.4. Команды выделения

Имя	Назначение
isolate(EQN,EX)	Определение выражения EX из уравнения EQN
denom(RAT)	Выделение знаменателя рациональной дроби RAT
numer(RAT)	Выделение числителя рациональной дроби RAT
lhs(EQN)	Выделение левой части из уравнения или диапазона EQN
rhs(EQN)	Выделение правой части уравнения EQN

Приведем простые примеры использования данных команд. Начнем с команды isolate для выделения одного из членов уравнения:

```
> isolate(x^2+a*y-x^4=1,a*y);
```

$a\,y = 1 - x^2 + x^4$

Однако при применении команды isolate нужно соблюдать осторожность. Введем уравнение:

```
> restart; eq:=x+y=x^2-y^2;
```

$eq := x + y = x^2 - y^2$

При попытке применения isolate для выделения выражения, которое не содержится явно, выводится сообщение о невозможности выполнения затребованной операции:

```
>  isolate(eq,x-y);
Error, (in isolate) x+y = x^2-y^2, does not contain, x-y
```

Попытка использовать команду isolate для выделения x+y приводит к некоторому результату, но не потеряно ли что-нибудь при такой процедуре?

```
> isolate(factor(eq),x+y);
```

$x + y = 0$

Для поиска неизвестных лучше использовать команды решения алгебраических уравнений и систем, которые описаны в главе 4 «Решение уравнений в Maple».

Рассмотрим ряд преобразований, позволяющих продемонстрировать команды выделения и найти решение уравнения eq. Образуем дробь, разделив правую часть уравнения eq на левую:

```
> rat:=rhs(eq)/lhs(eq);
```

$rat := \frac{x^2 - y^2}{x + y}$

Нормализуем дробь:

```
> rat:=normal(rat);
```

$rat := x - y$

Запишем выражение для разности величин x и y, определить которую не удалось прямым обращением к команде isolate:

```
> numer(rat)=denom(rat);
```

$x - y = 1$

Кстати, команда indets(EQ) позволяет получить список неизвестных в уравнении или системе EQ. Например, для введенного выше уравнения получим:

```
> indets(eq);
```

$\{x, y\}$

Заметим, что команды lhs и rhs полезны при выделении соответственно левой и правой части диапазона. Например, для использования в графической команде или операции интегрирования можно определить интервал:

```
> dia:=sqrt(Pi)..(a/b)^2:
```

Чтобы вычислить длину этого интервала, достаточно выполнить следующую команду:

```
> (rhs-lhs)(dia);
```

$$\frac{a^2}{b^2} - \sqrt{\pi}$$

Упрощение выражений — simplify

Действующий в Maple режим автоматического упрощения заключается в использовании информации о свойствах математических функций для преобразования выражений к более компактной форме. Это демонстрируют следующие простые команды:

```
> abs(abs(x)),arccos(sin(Pi/6)),GAMMA(1/2);
```

$$|x|, \frac{1}{3}\pi, \sqrt{\pi}$$

В более сложных ситуациях необходимо обращение к той или иной команде упрощения, а иногда и к последовательности операций преобразования. Команда simplify является мощным средством для проведения упрощающих преобразований, однако сам термин «упрощение» не вполне ясен. Получив выражение, Maple применяет ряд стандартных преобразований — factor, combine, expand и др., а также их комбинации, пытаясь получить итоговое выражение, которое было бы в некотором смысле проще исходного. Обращение к команде simplify имеет вид:

```
simplify(EX,OPT)
```

Упрощение выражения EX производится согласно заданным параметрам OPT, которые могут быть определены в виде некоторых соотношений, а также одним или несколькими терминами из следующего списка: @, Ei, exp, GAMMA, hypergeom, ln, polar, power, radical, RootOf, sqrt, trig и др. Например, при указании параметра trig упрощение производится в первую очередь с использованием тригонометрических соотношений. Если в качестве параметра OPT задать соотношения в виде равенств, то упрощение будет проводиться с учетом этих соотношений.

Рассмотрим простые примеры. Введем выражение с радикалом и попробуем его упростить, не привлекая дополнительной информации о подкоренных величинах:

```
> f:=sqrt(x^2*y^5): f=simplify(f);
```

$$\sqrt{x^2 y^5} = \sqrt{x^2 y^5}$$

Как видно, никакого упрощения не произошло. Используем соответствующий параметр, чтобы входящие в выражение переменные считались вещественными при упрощении:

```
> simplify(f,assume=real);
```

$$y^{(5/2)}|x|$$

Если указать, что переменные являются положительными, то получится:

```
> simplify(f,assume=positive);
```

$$x\,y^{(5/2)}$$

О команде assume будет рассказано в главе 3 «Математический анализ в Maple». Если в качестве дополнительного параметра использовать термин symbolic, то упрощение будет производиться без учета ограничений, накладываемых операцией (извлечение корня, логарифм и др.) на операнды:

```
> simplify(f,symbolic);
```

$$x\,y^{(5/2)}$$

При упрощении можно использовать дополнительные соотношения для переменных:

```
> simplify(f,{x*y=1});
```

$$\sqrt{y^3}$$

Помимо условий и соотношений можно определить переменные, относительно которых ведется упрощение. Переменные могут быть заданы в виде множества, тогда порядок перечисления переменных не важен, а для упорядоченности следует использовать список:

```
> f:=1-2*x^2*y^2+x^4*y+x^3*y^2-2*x*y^3-2*y^2:
simplify(f,{y^2=1+x,x*y=x^3-1},[y,x]);
```

$$-x^3-x^4-x^2$$

При другом порядке следования переменных для того же выражения получим ответ, в котором фигурируют только степени y:

```
> simplify(f,{y^2=1+x,x*y=x^3-1},[x,y]);
```

$$-1+y^2-y^5+y^3-y^4$$

При упрощении тригонометрических выражений бывает необходим ответ, выраженный через ту или иную функцию. Например, при упрощении тригонометрического выражения в ответе будут только косинусы:

```
> tri:=sin(x)^4-sin(x)^3*cos(x)+cos(x)^4-sin(x)*cos(x)^3
-1+sin(2*x)/2: simplify(tri);
```

$$2\cos(x)^4-2\cos(x)^2$$

Получить иное представление позволяет явное указание соответствующей функции:

```
> simplify(tri,{sin(x)})=simplify(tri,{cos(x)});
```

$$-1+\cos(x)^4=\sin(x)^4-1$$

Используя знаменитое тригонометрическое тождество, можно прийти к ответу следующего вида:

```
> eq:={sin(x)^2+cos(x)^2=1}:
simplify(tri,eq,{sin(x)})=simplify(tri,eq,{cos(x)});
```

$$2\cos(x)^4-2\cos(x)^2=2\sin(x)^4-2\sin(x)^2$$

Команда `simplify` эффективно упрощает многочлены и рациональные дроби, обрабатывает сложные выражения с радикалами. Отметим, что соответствующие средства Maple совершенствуются от версии к версии. Это видно хотя бы по тому, что не для всех примеров, приведенных в справке Maple, получение компактного ответа требует указанных в примерах параметров. Например, упрощение следующего выражения реализуется простым обращением к команде `simplify`:

```
> f:=x*(x-1)^(1/3)-(x-1)^(4/3):
simplify(f)=simplify(f,radical);
```

$$(x-1)^{(1/3)}=(x-1)^{(1/3)}$$

Работа с радикалами может потребовать привлечения дополнительных команд, которые можно найти в списке родственных команд (**See also**), завершающем описание команды или параметра в справке Maple. Приведем демонстрацию эффективности специальной команды нормализации выражений с радикалами `radnormal`:

```
> rad:=(2^(3/4)+2^(1/4))/(3*2^(1/2)+4)^(1/2):
simplify(rad)=radnormal(rad);
```

$$\frac{2^{(1/4)}(\sqrt{2}+1)}{\sqrt{3\sqrt{2}+4}}=1$$

Подстановка

Для замены переменной `OLD` выражением `NEW` в объект `EX` (выражение, список, множество, ...) применяется команда

```
subs (OLD=NEW,EX)
```

В одной команде разрешается использовать несколько выражений вида `OLD=NEW`, и этот набор подстановок может быть оформлен в виде множества.

Подстановка не изменяет исходный объект `EX`, а позволяет получить новое выражение. После операции подстановки иногда требуется одна из рассмотренных выше команд для упрощения или команда вычисления выражения (оценивания) `eval`, которая рассмотрена выше и также может использоваться для подстановок.

Например:

```
> s1:=subs(x=0,(sin(x)+1)*cos(x+Pi/6));
```

$$s1:=(\sin(0)+1)\cos\left(\frac{1}{6}\pi\right)$$

```
> factor(s2),expand(s2),simplify(s2),eval(s2);
```

$$\frac{1}{2}\sqrt{3},\frac{1}{2}\sqrt{3},\frac{1}{2}\sqrt{3},\frac{1}{2}\sqrt{3}$$

Чтобы такое оценивание производилось автоматически, можно вставить в файл инициализации Maple или ввести в текущем сеансе следующее определение макрокоманды (см. главу 7 «Программирование в Maple»):

```
> macro (subs=eval@subs):
```

В этом случае ответ выдается сразу:

```
> subs(x=0,s1);
```

$$\frac{1}{2}\sqrt{3}$$

При нескольких подстановках замещение производится слева направо. Это иллюстрируют следующие два примера:

```
> subs(x=y,z=x+y,z+x); subs(z=x+y,x=y,z+x);
```

$x+2\,y$

$3\,y$

Вместо последовательной подстановки можно использовать операцию одновременного замещения. Для этого подставляемые тождества нужно взять в фигурные скобки — превратить в множество. Это, например, позволяет поменять местами аргументы у функции нескольких переменных:

```
> subs({x=y,y=x},f(x,y));
```

$\mathrm{f}(y, x)$

С помощью команды subs можно заменять выражения. Однако при этом нужно, чтобы Maple распознал их. Образуем список, в котором произведение ab присутствует в разных видах, и применим к этому списку команду subs:

```
> ex:=[2*a*b+c,b*a+c*a*b,(a*b)^2]: subs(a*b=d,ex);
```

$[2\,d+c, d+a\,b\,c, a^2\,b^2]$

Пример показал, что подстановка командой subs выполнена не для всех случаев, что объясняется спецификой ее работы. Можно сказать, что команда выполняет «синтаксическое» замещение слов, не идентифицируя заменяемое, являющееся частью выражения, и не всегда обнаруживает все варианты. Для выполнения «математических» замещений, то есть включающих анализ выражений, имеется команда algsubs, которая легко распознает нужные величины в разных контекстах, но может оставить больше переменных, чем хотелось бы. Применение команды algsubs позволяет провести все подстановки в рассмотренном выше выражении:

```
> algsubs(a*b=d,ex);
```

$[2\,d+c, d+c\,d, d^2]$

Иногда требуется преобразовать выражение, выделив, например, произведение двух переменных. Непосредственное применение команды приведения подобных членов здесь не дает желаемого эффекта:

```
> ex:=a*b+c*a*b+1; collect(ex,a*b);
```

$ex := a\,b+c\,a\,b+1$

```
Error, (in collect) cannot collect a*b
```

Можно сначала выделить произведение, используя команду подстановки, а затем обратным замещением получить требуемое:

```
> ex1:=subs(a=ab/b,ex); ex2:=collect(ex1,ab);
```

$$ex1 := ab + c\,ab + 1$$

$$ex2 := (1 + c)\,ab + 1$$

```
> ex3:=subs(ab=a*b,ex2);
```

$$ex3 := (1 + c)\,a\,b + 1$$

Конечно, простые подстановки можно делать при помощи оператора присваивания, когда переменной присваивается некоторое выражение. Однако в этом случае введенная подстановка распространится и на следующие операции с участием этой переменной.

Команда подстановки subs не позволяет проводить замену в выражении производной, не действует под знаком многократного интеграла, в выражении предела и некоторых других случаях; для таких подстановок нужно применять команду dchange из пакета PDEtools.

Отметим также интересную команду trigsubs(EX), позволяющую, в частности, вывести все тригонометрические эквиваленты выражения EX. Например, для синуса двойного угла получим:

```
> trigsubs(sin(2*x));
```

$$\left[\sin(2\,x), -\sin(-2\,x), 2\sin(x)\cos(x), \frac{1}{\csc(2\,x)}, -\frac{1}{\csc(-2\,x)}, 2\,\frac{\tan(x)}{1+\tan(x)^2}, \frac{-1}{2}\,I\,(e^{(2\,I\,x)} - e^{(-2\,I\,x)})\right]$$

Операции с полиномами

Для всех пакетов аналитических вычислений операции с полиномами являются базовыми и часто используются при других преобразованиях формул. Под полиномом в Maple понимается сумма выражений одной или нескольких переменных с неотрицательными степенями, так что полиномами являются константа, простая переменная и выражение.

Рассмотрим представительное подмножество команд работы с полиномами. В табл. 2.5 приведен список команд с короткими пояснениями. Здесь использованы следующие обозначения: P, P1, P2 — полиномы, VAR — переменная или список переменных, N — целое число (степень, порядок), NAME — имя для присвоения результата операции.

Рассмотрим следующий пример: введем полином p двух переменных x и y, умножим его на выражение x+2, используем команду expand для раскрытия скобок и результат присвоим переменной q:

```
> p:=x-2*y+x*y-2; q:=expand(p*(x+2));
```

$$p := x - 2\,y + x\,y - 2$$

$$q := x^2 - 4\,y + x^2\,y - 4$$

Таблица 2.5. Команды работы с полиномами

Команда	**Назначение**
coeff(P,VAR,N)	Вычисление коэффициента при N-й степени переменной VAR для полинома P
coeffs(P,VAR)	Вычисление коэффициентов полинома P по переменной (переменным) VAR
degree(P,VAR)	Вычисление степени полинома P по переменной VAR
discrim(P,VAR)	Вычисление дискриминанта полинома P относительно переменной VAR
divide(P1,P2,NAME)	Вычисление частного от деления двух полиномов
gcd(P1,P2)	Вычисление наибольшего общего делителя двух полиномов
lcoeff(P,VAR)	Вычисление старшего коэффициента полинома
ldegree(P,VAR)	Вычисление наименьшей степени полинома относительно переменной VAR
proot(P,N)	Нахождение корня N-й степени из полинома P
psqrt(P)	Нахождение квадратного корня из полинома P
quo(P1,P2,VAR)	Вычисление частного от деления двух полиномов
rem(P1,P2,VAR)	Вычисление остатка от деления двух полиномов
randpoly(VAR)	Создание полинома переменной (переменных) VAR со случайными коэффициентами
realroot(P)	Вычисление интервала, где лежат вещественные корни полинома
tcoeff(P,VAR)	Вычисление свободного члена полинома P

При помощи команды divide разделим получившееся выражение на полином y+1:

```
> divide(q,y+1,p1);
```

true

Отметим, что для команды divide выводится булева величина (true при успешном делении и false – при неудаче), а результат присваивается переменной, являющейся третьим параметром команды. Выведем полученный в результате деления полином p1, определим его степень и коэффициент при старшей степени:

```
> p1; degree(p1), lcoeff(p1);
```

x^2-4

2, 1

Наконец, при помощи команды gcd найдем наибольший общий делитель полиномов p1 и x^2-2x,

```
> gcd(p1,x^2-2*x);
```

$x-2$

Результатом выполнения команды coeffs будет переменная типа list, содержащая неупорядоченные коэффициенты. Для получения соответствия между коэффициентами и степенями переменной VAR следует использовать расширенный формат команды с указанием третьего параметра:

```
> coeffs(x^2+y*x-x^4+y^2+x,x,"s"); s;
```

$y^2, -1, 1, y+1$

$1, x^4, x^2, x$

Итак, в переменной s степени идут в порядке, соответствующем перечислению коэффициентов.

Теперь приведем пример извлечения корня из полинома:

```
> p:=x^6+2*x+1-x^2-x^4-4*x^3+2*x^5; psqrt(p)=proot(p,2);
```

$p := -x^2 - x^4 - 4\,x^3 + 2\,x^5 + x^6 + 2\,x + 1$

$-1 - x + x^2 + x^3 = (x-1)\,(1+x)^2$

При невозможности извлечь корень появляется сообщение:

```
> proot(p,3);
```

_NOROOT

Напомним, что для разложения полинома P на множители достаточно выполнить команду factor(P).

Обратим внимание на специфические приложения команды convert для операций с полиномами: convert(P,horner,VAR) — преобразование полинома по схеме Горнера, convert(P,mathorner,VAR) — преобразование полинома по матричной форме схемы Горнера, convert(P,sqrfree,VAR) — разложение полинома на квадратные трехчлены.

Приведем пример:

```
> p:=randpoly([x,y],degree=2);
```

$p := 56 + 49\,x + 63\,y + 57\,x^2 - 59\,x\,y + 45\,y^2$

```
> p1:=convert(p,horner,[x,y]);
```

$p1 := 56 + (63 + 45\,y)\,y + (-59\,y + 49 + 57\,x)\,x$

```
> P2:=subs(y=0,x=array([[2,1],[1,3]]),eval(p1)); evalm(P2);
```

$$P2 := 56 + \left(49 + 57\begin{bmatrix}2 & 1\\1 & 3\end{bmatrix}\right)\begin{bmatrix}2 & 1\\1 & 3\end{bmatrix}$$

$$\begin{bmatrix}439 & 334\\334 & 773\end{bmatrix}$$

Для сортировки полиномиальных выражений применяется та же команда sort, которая позволяет упорядочивать элементы списка, см. главу 1 «Основы Maple». Это связано со способом хранения полиномов — в виде списка. По умолчанию члены полинома располагаются по убывающим степеням, и после применения команды sort полином хранится в преобразованном виде. Для полинома от нескольких переменных при помощи второго параметра команды sort — списка или множества переменных — можно указать желаемый порядок следования членов с переменными. Если указать параметр "plex" (можно использовать и апострофы), то сортировка будет производиться в лексикографическом порядке.

Приведем ряд примеров. Введем полином трех переменных b, x, y и выполним команду сортировки без дополнительных параметров. В результате получим:

```
> po:=x*y*b+x*y^3-x^3+y*x^2; sort(po);
```

$po := y^3\,x + y\,x^2 - x^3 + b\,y\,x$

$x\,y^3 - x^3 + x^2\,y + x\,y\,b$

Теперь посмотрим, как выполнится сортировка при указании переменной, по степеням которой следует расположить члены полинома:

```
> sort(po,x,"plex"); sort(po,y);
```

$-x^3 + y\,x^2 + y^3\,x + y\,b\,x$

$x\,y^3 + x^2\,y + b\,x\,y - x^3$

Опробуем две комбинации переменных, по которым нужно производить сортировку:

```
> sort(po,[y,x]); sort(po,[b,x],"plex");
```

$y^3\,x + y\,x^2 - x^3 + b\,y\,x$

$y\,b\,x - x^3 + y\,x^2 + y^3\,x$

Наконец, убедимся, что полином записан согласно последней команде сортировки:

```
> po;
```

$y\,b\,x - x^3 + y\,x^2 + y^3\,x$

Ряд команд для работы с полиномами содержится в небольшой библиотеке polytools, которая загружается целиком по команде:

```
> with(polytools);
```

[*minpoly*, *recipoly*, *split*, *splits*, *translate*]

При вызове одной команды COM можно использовать полное имя: polytools[COM]() или подгрузить команду при помощи вызова with(polytools,COM).

Команда minpoly(R,N,ACC) позволяет построить полином с целыми коэффициентами степени не выше N, такой, что его корнем является заданное значение R.

При помощи дополнительного параметра ACC указывается точность, по умолчанию ACC равно 10^(Digits-2), здесь Digits — переменная среды (константа) Maple, определяющая число значащих цифр в вычислениях с плавающей запятой.

Найдем полином первой степени с корнем, приближающим число π:

```
> po:=minpoly(Pi,1);
```

$po := -355 + 113\,_X$

Воспользуемся командой roots для определения корня и его кратности. Преобразуем дробь в десятичное число и убедимся, что полученное значение содержит семь верных знаков числа π:

```
> roots(po,_X); evalf(%[]);
```

$\left[\left[\frac{355}{113}, 1\right]\right]$

[3.141592920, 1.]

Помимо полиномов общего вида имеются полиномы, носящие имена великих математиков — Лежандра, Чебышева, Якоби и др. Эти полиномы и библиотека ортогональных полиномов `orthopoly` описаны в главе 8 «Математические библиотеки Maple».

В версиях, предшествующих Maple 6, перед запуском ряда команд требовалось их подгрузить из стандартной библиотеки при помощи команды `readlib`. Это касалось, например, рассмотренных здесь команд `proot` и `psqrt`.

3 ГЛАВА Математический анализ в Maple

В этой главе изложены команды Maple, реализующие основные математические операции, известные читателям в основном из математического анализа и высшей математики в вузе. Мы предполагаем, что человек, взявший в руки эту книгу, имеет за плечами по крайней мере один курс университета или технического вуза и, следовательно, знаком с основами математического анализа.

Работу команд Maple в этом разделе мы будем иллюстрировать примерами из известных задачников Демидовича [50, 51], Кудрявцева [59]. Перед тем как перейти к последовательному изложению математических команд, сообщим ряд необходимых сведений.

Предварительные сведения

В Maple для некоторых математических операций существует по две команды: одна прямого, а другая — отложенного исполнения, причем имена этих команд состоят из одинаковых букв. Команды прямого исполнения, как правило. начинаются с маленькой буквы и выполняются немедленно. Отложенные команды часто начинаются с большой буквы, обычно в том случае, когда существует команда-синоним прямого действия. После обращения к команде отложенного действия заданная математическая операция (интеграл, производная, предел и т. д.) выводится в стандартном математическом виде и сразу не вычисляется. Для выполнения отложенной операции нужно использовать команду `value`. Перед использованием некоторых команд необходимо предварительно подключить соответствующую библиотеку командой `with`.

Напомним, что результат последней выполненной операции сохраняется в переменной `%`, предпоследней команды — в переменной `%%`, а ей предшествующей — `%%%`.

Часто необходимо наложить условия на переменные, например для задания области определения функции; в Maple это можно сделать, используя команду

```
assume (rel,prop)
```

Здесь rel — выражение, а prop — свойства. В качестве свойств могут выступать: integer, real, continuous, RealRange(a,b) и другие (см. справку Maple). Например, после команды

```
> assume(b,integer);
```

переменная b будет далее считаться целым числом, а после команды

```
> assume(a>0);
```

везде, где будет встречаться переменная a, она будет обозначаться a~ и считаться положительной:

```
> Re(a); Im(a);
```

a~

0

Для проверки условий, наложенных на переменные, и добавления новых свойств переменных существуют команды:

- about(x) — выводит информацию об условиях, наложенных на переменную x;
- additionally(x,prop) — накладывает дополнительные условия на переменную x;
- addproperty(prop,parents,children) — определяет новое свойство prop, основанное на свойствах parents и children;
- coulditbe(expr,prop) — проверяет, может ли выражение expr удовлетворять свойству prop. Результатом является булевская константа true или false;
- is(x,prop) — проверяет, удовлетворяет ли выражение expr свойству prop. Результатом является булевская константа true или false.

Продолжим пример с определением свойств переменной a, используя описанные команды:

```
> about(a);
Originally a, renamed a~:
  is assumed to be: RealRange(Open(0),infinity)
> coulditbe(a*(-a) < -200);
```

true

```
> is(a^2,negative);
```

false

Чтобы отменить назначения для переменной name, используется команда

```
name:='name';
```

Например:

```
> a:='a': about(a);
a:
  nothing known about this object
```

Пределы, суммы, ряды

Обычно курс математического анализа начинается с понятия предела последовательности и функции. Для вычисления пределов в Maple существуют команды

```
limit (expr,x=val,dir)
```

и

```
Limit (expr,x=val,dir)
```

Здесь expr — выражение, для которого вычисляется предел (функция или n-й член последовательности), x=val — точка, в которой вычисляется предел, а dir — необязательный параметр, который может принимать следующие значения: left (предел слева), right (предел справа), real (действительный) или complex (комплексный). Проиллюстрируем сказанное примерами:

```
> Limit(n*sin(n!)/(n^2+1),n=infinity); value(%);
```

$$\lim_{n \to \infty} \frac{n \sin(n!)}{n^2+1}$$

$$0$$

```
> limit(ln(n*x+sqrt(1-n^2*x^2))/ln(x+sqrt(1-x^2)),x=0);
```

$$n$$

```
> Limit(1/(1+exp(1/x)),x=0,left)=
       limit(1/(1+exp(1/x)),x=0,left);
```

$$\lim_{x \to 0-} \frac{1}{1+e^{\left(\frac{1}{x}\right)}} = 1$$

```
> limit(1/(1+exp(1/x)),x=0,right);
```

$$0$$

Для операции суммирования используются команды

```
sum (expr,var=var1..var2)
```

и

```
Sum (expr,var=var1..var2)
```

Здесь expr — выражение, зависящее от переменной суммирования var, а var1..var2 — пределы суммирования. Напомним, что различие между этими двумя командами заключается в том, что в варианте, начинающемся с маленькой буквы, суммирование проводится сразу, а для выполнения Sum нужно дополнительно дать команду value. Пределы суммирования могут быть конечными, бесконечными или отсутствовать. Эта команда может быть использована также и для суммирования рядов. Например:

```
> sum(2^n*(n+1)/n!,n=0..30);
```

$$\frac{8761740900072755414912520 1}{3952575621190533915703125}$$

```
> sum(n*(n+2)*x^n,n=1..infinity);
```

$$-\frac{x(-x+3)}{(x-1)^3}$$

```
> Sum((n+2)*x^n,"n"); %=value(%);
```

$$\sum_{n} (n+2)\, x^n = \frac{x^n (x-2+n\,x-n)}{(x-1)^2}$$

Для вычисления бесконечных и конечных произведений используются две команды:

```
product (expr,k)
```

и

```
Product (expr,k=k1..k2)
```

Здесь expr — k-й член произведения, k — индекс, а k1 и k2 задают интервал изменения индекса. Например:

```
> Limit(Product(1-1/n^2,n=2..k),k=infinity): %=value(%);
```

$$\lim_{k \to \infty} \prod_{n=2}^{k} \left(1 - \frac{1}{n^2}\right) = \frac{1}{2}$$

В пакете student существуют одноименные команды Sum, Limit и Product, которые позволяют производить выкладки с этими объектами. Например:

```
> with(student): a:=Sum(1/x+1/x^2,"x");
```

$$a := \sum_{x} \left(\frac{1}{x} + \frac{1}{x^2}\right)$$

```
> expand(a); value(%);
```

$$\left(\sum_{x} \frac{1}{x}\right) + \left(\sum_{x} \frac{1}{x^2}\right)$$

$$\Psi(x) - \Psi(1, x)$$

Исследование, разложение и приближение функций

Как известно, исследование функции необходимо начинать с выяснения ее области определения. К сожалению, это трудно автоматизируемая операция и решение данной задачи не может быть пока полностью переложено на плечи Maple. Обычно при рассмотрении этого вопроса приходится решать неравенства, с чем пакет довольно успешно справляется (см. раздел о решении уравнений).

Проверить непрерывность алгебраического выражения, зависящего от переменной x, на отрезке [x1,x2] можно при помощи команды

```
iscont (expr,x=x1..x2)
```

Результатом выполнения команды будет булевская константа «истина» (true) или «ложь» (false). Для определения точек, в которых нарушается непрерывность выражения expr по переменной x, используется команда

```
discont (expr,x)
```

Для определения координат, где возникает разрыв функции f или ее первой производной, есть еще одна команда:

```
fdiscont (f,domain,res,var,opt)
```

Здесь domain — изучаемая область, res — точность, var — имя независимой переменной, opt — параметры (см. справку пакета).

Приведем пример:

```
> iscont(ln(cos(x)),x = -infinity..+infinity);
```

false

```
> discont(ln(cos(x)),x);
```

$$\{\frac{1}{2}\pi + \pi_Z2\sim\}$$

Здесь _Z2~ означает любое целое положительное число.

Кроме того, чтобы найти точки разрыва функции, можно обратиться к команде

```
singular (expr,vars)
```

Здесь expr — выражение, зависящее от переменных vars, причем expr может зависеть и от других переменных, а точки разрыва ищутся для переменных vars. Например:

```
> singular(tan(x/(x-y)),x);
```

$$\{x = y\}, \{x = \frac{\pi\, y\,(2\,_Z3\sim + 1)}{-2 + 2\,_Z3\sim\,\pi + \pi}\}$$

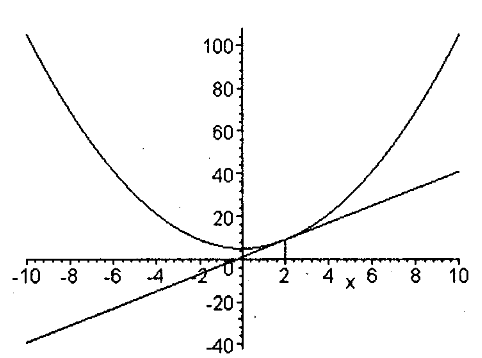
Здесь _Z3 означает любое целое число.

В пакете student существует несколько команд для анализа функций:

- showtangent (f(x),x=a) — показывает графически касательную к функции f(x) в точке x=a;
- intercept(eqn1, eqn2, {x, y}) — находит точку пересечения двух кривых, заданных уравнениями eqn1 и eqn2.

Пример:

```
> student[showtangent](x^2+5, x = 2);
```



Для исследования экстремумов функции одной и многих переменных используется команда

```
extrema (expr,constr, vars,'nv')
```

Здесь expr — выражение, экстремумы которого нужно найти, constr — ограничения, vars — переменные, по которым разыскивается экстремум, а nv — имя переменной, которой будут присвоены координаты точек экстремумов. Соответствующий пример:

```
> extrema(arctan(x)-ln(1+x^2)/2, {} ,x,'s'); s;
```

$$\{\frac{1}{4}\pi-\frac{1}{2}\ln(2)\}$$

$$\{\{x=1\}\}$$

Для поиска минимума и максимума функции или выражения используются соответственно команды

```
minimize (expr,vars,ranges,opts)
```

и

```
maximize (expr,vars,ranges,opts)
```

Здесь expr — выражение, vars — переменные, по которым ищется минимум или максимум, а ranges — область изменения переменных, причем здесь может стоять строка "infinite", то есть минимум или максимум будет разыскиваться на всей числовой оси. Одним из возможных значений необязательных параметров opts является location, что указывает на необходимость вывода координат минимума или максимума. Пример:

```
> minimize(x^2-3*x+y^2+3*y+3, x=2..4, y=-4..-2, location);
```

$$-1, \{[\{x=2, y=-2\}, -1]\}$$

Вычет алгебраического выражения expr при x=a вычисляется при помощи команды

```
residue (expr,x=a)
```

Теперь перейдем к командам разложения функций в ряды. При этих операциях порядок разложения по умолчанию определяется значением зарезервированной константы Order, которая может переопределяться пользователем в процессе работы.

Команда series(expr, eqn, n) выполняет разложение выражения expr в степенной ряд; здесь eqn — выражение вида x=a, где x — переменная разложения, a — точка, в окрестности которой выписывается разложение, n — порядок, до которого строится разложение. Приведем соответствующий пример:

```
> series((1+x)^(1/x),x=0,4);
```

$$e-\frac{1}{2}e\,x+\frac{11}{24}e\,x^2+O(x^3)$$

У команды series есть еще один вариант, который позволяет получить ведущий член степенного разложения выражения expr. Он имеет вид:

```
series("leadterm"(expr), x=a, n)
```

Приведем пример:

```
> series("leadterm"(sqrt(sin(x))),x=0);
```

$$\sqrt{x}$$

Для разложения выражения expr, зависящего от переменной x, в ряд Тейлора в точке x=a до членов порядка n, можно воспользоваться командой

```
taylor (expr,x=a,n)
```

Отметим, что если последний параметр не указан, то порядок разложения определяется значением константы Order.

```
> taylor(sin(x^3)^(1/3),x=0,24);
```

$$x - \frac{1}{18}x^7 - \frac{1}{3240}x^{13} - \frac{53}{1224720}x^{19} + O(x^{25})$$

Выражения, зависящие от нескольких переменных, разлагаются в ряд Тейлора при помощи команды

```
mtaylor (expr,vars,k,w)
```

Здесь expr — разлагаемое в ряд выражение, vars — список пар <имя переменной> =<точка> для переменных разложения, k — порядок разложения, а w — вес. Проиллюстрируем действие команды на следующем примере:

```
> mtaylor(exp(x)*cos(y),[x=ln(a),y=Pi],3);
```

$$-a - a\,(x - \ln(a)) + \frac{1}{2}a\,(y-\pi)^2 - \frac{1}{2}a\,(x - \ln(a))^2$$

Практически аналогична предыдущей по своему действию команда poisson(f,v,n,w) и отличается представлением тригонометрических функций в разложении.

Если нужно получить только коэффициент при члене порядка k, то лучше воспользоваться командой

```
coeftayl (expr,vars,k)
```

Здесь vars — список имен переменных разложения, k — порядок члена разложения, при котором выписывается коэффициент. Для функции нескольких переменных параметр k является списком.

Для разложения выражения expr в ряд Лорана можно использовать команду laurent(expr,x=a,n) из пакета numapprox с параметрами, которые аналогичны параметрам команды taylor. Команда asympt(f,x,n) позволяет выписать асимптотическое разложение выражения f по степеням переменной x порядка n, когда x стремится к бесконечности.

Приближенные аналитические вычисления

Результатом разложения выражения в ряд является переменная типа series (формальное степенное разложение). Покажем это:

```
> f_s:=taylor(ln(1+x),x,5);
```

$$f_s := x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + O(x^5)$$

```
> whattype(f_s);
```

series

Для преобразования переменной ser типа series в алгебраическое выражение применяется команда

```
convert (ser,polynom)
```

Для работы с формальными степенными разложениями существует пакет powseries. Команды этого пакета позволяют задавать разложения как посредством рекуррент-

ных формул (powcreate), так и через обычные полиномы (powpoly), а также проводить с ними различные операции. Для хранения разложений используется специальный формат, поэтому для просмотра их нужно переводить в обычные ряды при помощи команды (tpsform).

Коротко опишем имеющиеся в пакете команды. Определяемым пользователем степенным разложениям присваивается имя, которое далее используется в аргументах следующих команд: вычисления суммы (add) и разности (subtract), умножения на заданное выражение (multconst), произведения (multiply) и деления (quotient), дифференцирования (powdiff) и интегрирования (powint), вычисления экспоненты (powexp) и натурального логарифма (powlog), вычисления мультипликативного (inverse) и аддитивного (negative) обратных, композиции (compose) и обратного к композиции разложения (reversion).

Имеется также аналог команды eval — команда evalpow, позволяющая вычислять сложные выражения, в которых участвуют разложения и используются обычные знаки для сложения, вычитания и умножения, а также специальные операции. Для решения линейного дифференциального уравнения в виде формального разложения применяется команда powsolve.

Приведем демонстрационный пример. Вначале подключим пакет и определим рекуррентно разложение функции ln(1+x), которому будет присвоено имя f:

```
> with(powseries): powcreate(f(n)=a^n/n!);
> f_s:=tpsform(f,x,5);
```

$$f_s := 1 + a\,x + \frac{1}{2}a^2x^2 + \frac{1}{6}a^3x^3 + \frac{1}{24}a^4x^4 + O(x^5)$$

Следующей командой найдем обратное разложение и умножим его на исходное выражение:

```
> finv:=inverse(f): tpsform(finv,x,5);
```

$$1 - a\,x + \frac{1}{2}a^2x^2 - \frac{1}{6}a^3x^3 + \frac{1}{24}a^4x^4 + O(x^5)$$

```
> p:=multiply(finv,f): tpsform(p,x,35);
```

$$1 + O(x^{35})$$

Теперь продемонстрируем работу пакета на примере построения формального разложения для решения дифференциального уравнения. Подробнее различные методы решения обыкновенных дифференциальных уравнений обсуждаются в соответствующей главе. Обратим внимание, что при постановке начальных условий отсутствовало условие для второй и третьей производных и в ответе возникли константы C2 и C3:

```
> eq:=diff(y(x),x$5)=-y(x),y(0)=0,D(y)(0)=-1;
```

$$eq := \frac{\partial^5}{\partial x^5}\,y(x) = -y(x),\ y(0) = 0,\ D(y)(0) = -1$$

```
> v:=powsolve({eq,D(D(D(D(y))))(0)=5}): tpsform(v,x,7);
```

$$-x + \frac{1}{2}C2\,x^2 + \frac{1}{6}C3\,x^3 + \frac{5}{24}x^4 + \frac{1}{720}x^6 + O(x^7)$$

Аппроксимация функций

При помощи Maple можно решать задачи приближения функций. Например, для разложения выражения expr в ряд по полиномам Чебышева относительно переменной x с точностью eps в пакете **numapprox** существует команда chebyshev(expr,x,eps). В этом пакете имеется еще ряд команд для аппроксимации выражений рациональными полиномами:

- chebpade — вычисление аппроксимации и паде-аппроксимации при помощи полиномов Чебышева;
- confracform — преобразование рациональной функции в цепную дробь;
- hornerform — преобразование полинома по схеме Горнера;
- infnorm — вычисление нормы L_∞;
- minimax — вычисление наилучшей минимаксной рациональной аппроксимации;
- pade — вычисление паде-аппроксимации;
- remez — вычисление наилучшей минимаксной рациональной аппроксимации по алгоритму Ремеза.

Параметры команд для краткости мы опустили, но их легко посмотреть в справке Maple. Приведем примеры обращения к командам пакета, выбрав синус в качестве аппроксимируемой и оцениваемой функции. Вначале получим обычное разложение в ряд Лорана и превратим его в рациональную дробь:

```
> with(numapprox):
> c:=laurent(sin(x),x); convert(c,ratpoly);
```

$$c := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + O(x^6)$$

$$\frac{-\frac{7}{60}x^3 + x}{1 + \frac{1}{20}x^2}$$

Затем вычислим паде-аппроксимацию и преобразуем ее в цепную дробь:

```
> p:=pade(sin(x),x,[4,4]); pc:=confracform(p);
```

$$p := \frac{-\frac{31}{294}x^3 + x}{1 + \frac{3}{49}x^2 + \frac{11}{5880}x^4}$$

$$pc := -\frac{620}{11}\,\frac{1}{x + \cfrac{\frac{14394}{341}}{x - \frac{1647086}{74369}\,\cfrac{1}{x + \cfrac{\frac{30380}{2399}}{x}}}}$$

Вычислим норму для синуса на заданном отрезке и найдем точку, в которой достигается максимум:

```
> infnorm(sin,0..Pi, "xmax"); xmax;
```

1.000000000

1.570796248

Проведем аналогичное вычисление для паде-аппроксимации синуса; при этом используем команду unapply для превращения выражения в функцию, так как команда infnorm оценивает норму функции:

```
> infnorm(unapply(p,x),0..Pi, "xmax"); xmax;
```

.9997177605

1.569270436

Далее обратимся к команде minimax для получения наилучшей аппроксимации синуса полиномом второй степени:

```
> minimax(sin(x),x=0..Pi,2);
```

$-.0280040553 + (1.273239545 - .4052847348\ x)\ x$

Наконец, используем для аппроксимации полиномы Чебышева, причем для сокращения числа цифр уменьшим мантиссу:

```
> Digits:=5: ch:=chebpade(sin(x),x,5);
```

$ch := .88010\ \mathrm{T}(1, x) - .039127\ \mathrm{T}(3, x) + .00049952\ \mathrm{T}(5, x)$

Теперь сравним значения двух приближений с аппроксимируемой функцией в точке $\pi/2$, преобразовав выражения в функции и подключив пакет вычисления ортогональных полиномов:

```
> Digits:=10: f1:=unapply(ch,x): f2:=unapply(p,x):
with(orthopoly):
> evalf(f1(Pi/2)-sin(Pi/2)); evalf(f2(Pi/2)-sin(Pi/2));
```

.001882256

-.0002834124

Дифференцирование и интегрирование

Вычисление производной выражения expr по переменной x осуществляется при помощи команд diff(expr,x) или Diff(expr,x). Отметим, что эти команды могут использоваться и для вычисления частных производных функций многих переменных, в этом случае используется следующий формат:

```
diff(expr,x1$n1,x2$n2,...)
```

Здесь expr — выражение, зависящее от переменных x1,x2, ..., а n1,n2, ... — порядки дифференцирования по соответствующим переменным. Для команды Diff параметры аналогичны. Операция diff используется и во многих других случаях (например, см. раздел «Обыкновенные дифференциальные уравнения» в главе 4 «Решение уравнений в Maple»). Приведем примеры:

```
> Diff(ln(tan(x/2))/2-cos(x)/sin(x)^2/2,x):
```

```
%=simplify(value(%));
```

$$\frac{\partial}{\partial x}\left(\frac{1}{2}\ln\left(\tan\left(\frac{1}{2}x\right)\right)-\frac{1}{2}\frac{\cos(x)}{\sin(x)^2}\right)=\frac{\sin(x)}{(-1+\cos(x)^2)^2}$$

```
> u:=z^(x*y): diff(u,x); diff(u,y); diff(u,z);
```

$$z^{(x\,y)}\,y\ln(z)$$

$$z^{(x\,y)}\,x\ln(z)$$

$$\frac{z^{(x\,y)}\,x\,y}{z}$$

```
> factor(diff(u,z$4));
```

$$\frac{z^{(x\,y)}\,x\,y\,(x\,y-3)\,(x\,y-1)\,(x\,y-2)}{z^4}$$

Если правило дифференцирования функции пакету неизвестно или одна из переменных неявно зависит от другой, то в ответе будут сохраняться символы дифференцирования. Пример:

```
> alias (y=y(x)): diff(x^2+y^2=g(x),x);
```

$$2\,x+2\,y\left(\frac{\partial}{\partial x}y\right)=\frac{\partial}{\partial x}\mathrm{g}(x)$$

В Maple существует возможность определять с помощью своей процедуры (о процедурах см. главу 7 «Программирование в Maple») правила дифференцирования для функций. Поясним эту возможность примером, в котором мы определим правило дифференцирования функции `f`:

```
> "diff/f" := proc(g,x) diff(g,x)*f(x) end proc:
  diff(f(sin(x)),x);
```

$$\cos(x)\,\mathrm{f}(x)$$

Для задания дифференциального оператора используется символ `D`. Чтобы преобразовать выражение с оператором `D` к виду, использующему команду `diff`, применяется команда `convert`. Она же используется и для обратного преобразования. Поясним сказанное следующими примерами:

```
> D(sin); D(ln-arcsin);
```

$$\cos$$

$$\left(a\rightarrow\frac{1}{a}\right)-\left(a\rightarrow\frac{1}{\sqrt{1-a^2}}\right)$$

```
> f:=diff(y(x),x$7); convert(f,D);
```

$$f:=\frac{\partial^7}{\partial x^7}\mathrm{y}(x)$$

$$(\mathrm{D}^{(7)})(y)(x)$$

Операторное дифференцирование `D` применяется в тех случаях, когда ищется производная функции, а не выражения, и результатом действия оператора будет также функция. Пример:

```
> g:=x->x*sin(x);
```

$g := x \to x \sin(x)$

```
> gd:=D(g); gd(Pi);
```

$gd := x \to \sin(x) + x\cos(x)$

$-\pi$

Аналогичного результата можно добиться при помощи команды unapply(diff(g(x),x),x).

Для вычисления частных производных функции нескольких переменных используется обращение к оператору D с индексами, которые указывают, по какому аргументу дифференцировать. Например, вычисление оператора Лапласа L функции h будет выглядеть следующим образом:

```
> h:=(x,y)->sin(x)*cos(y):
  L:=(D[1,1]+D[2,2])(h);
```

$L := ((x, y) \to -\sin(x)\cos(y)) + ((x, y) \to -\sin(x)\cos(y))$

Оператор D можно применять к любым Maple-процедурам (о процедурах см. в главе 7 «Программирование в Maple»), например:

```
> fun:=proc(x) if x<0 then sin(x)
           else ln(x) end if; end proc:
dfun:=D(fun);
```

dfun := proc (*x*) if *x* < 0 then cos(*x*) else 1/*x* end if end proc

Теперь перейдем к интегрированию функций. Для этой цели в пакете предусмотрено несколько команд, находящихся в различных библиотеках. В стандартной библиотеке находятся процедуры int(expr,par) и Int(expr,par), которые в зависимости от параметров par могут использоваться для поиска неопределенных интегралов, аналитического или численного вычисления определенных, собственных и несобственных интегралов. Для поиска неопределенных интегралов используется следующая команда:

```
int(expr,var)
```

Здесь expr — интегрируемое выражение, а var — переменная интегрирования. Для отложенной команды Int все аналогично. Пример:

```
> int((1+sin(x))*exp(x)/(1+cos(x)),x);
```

$$e^x \tan\left(\frac{1}{2}x\right)$$

Для вычисления определенных интегралов используется команда

```
int(expr,var=a..b)
```

Здесь expr — интегрируемое выражение, var — переменная интегрирования, а (a,b) — отрезок интегрирования, его концы могут принимать значения бесконечности. Для вычисления двойных, тройных и т. д. интегралов нужно применить эту команду несколько раз. Поясним это примерами:

```
> Int(sinh(x)^4,x=0..ln(2))=int(sinh(x)^4,x=0..ln(2));
```

$$\int_0^{\ln(2)} \sinh(x)^4\,dx = -\frac{225}{1024} + \frac{3}{8}\ln(2)$$

Теперь вычислим повторный интеграл:

```
> Int(Int(x+2*y,x=y^2-4..5),y=-3..3): %=value(%);
```

$$\int_{-3}^{3}\int_{y^2-4}^{5} x+2\,y\,dx\,dy=\frac{252}{5}$$

Если интеграл не выражается через элементарные функции, то пакет пытается получить ответ в специальных функциях. Иногда ответ удается выразить через элементарные функции, используя дополнительные условия на параметры и переменные (команда assume). Проиллюстрируем сказанное примером:

```
> Int(exp(-c*x^2),x=0..infinity): %=value(%);
Definite integration: Can't determine if the integral is convergent.
Need to know the sign of -> c
Will now try indefinite integration and then take limits.
```

$$\int_{0}^{\infty} e^{(-c\,x^2)}\,dx=\lim_{x\to\infty}\frac{1}{2}\frac{\sqrt{\pi}\,\mathrm{erf}(\sqrt{c}\,x)}{\sqrt{c}}$$

Без дополнительных предположений интеграл выразился через предел функции ошибок. Теперь предположим, что параметр c строго положителен, и повторим вычисления:

```
> assume (c>0): Int(exp(-c*x^2),x=0..infinity): %=value(%);
```

$$\int_{0}^{\infty} e^{(-c\sim x^2)}\,dx=\frac{1}{2}\frac{\sqrt{\pi}}{\sqrt{c\sim}}$$

Если интеграл не удается взять аналитически и подынтегральное выражение не содержит неопределенных параметров, то его можно вычислить численно. Для этого предназначена команда

```
evalf(int (expr,x=a..b,digits,flag))
```

Здесь обязательными параметрами являются подынтегральная функция expr, зависящая только от переменной x и пределов интегрирования a и b, а необязательными — число значащих цифр digits (по умолчанию принимает значение константы Digits) и flag — код численного метода. Пример:

```
> Int( sin(x)*ln(x), x = 0..1 ):
> value(%)=evalf(Int( sin(x)*ln(x), x = 0..1 ,15,_NCrule));
```

$$\mathrm{Ci}(1)-\gamma=-.239811742000565$$

Здесь Ci — интегральный косинус, а γ — константа Каталана. Несколько команд интегрирования имеется в библиотеке **student**. Отметим, что все эти команды отложенного действия и библиотеку предварительно нужно подключить при помощи with(student). Кратко их перечислим:

- Int(expr,x) — интегрирование выражения expr по переменной x;
- Doubleint(expr,x,y,Domain) — двойное интегрирование выражения expr по переменным x и y в области Domain;

- Lineint(f(x,y),x,y) — вычисление линейного интеграла. Переменная x считается зависящей от переменной y, а если переменных больше, то все они считаются зависящими от последней;
- Tripleint(expr,x,y,z) — вычисление тройного интеграла;
- changevar(s,f,u) — замена переменных, где s — выражение, задающее замену координат, f — одна из перечисленных команд интегрирования из пакета student, а u — список новых переменных интегрирования;
- intparts(f,u) — интегрирование по частям, где f — выражение Int[student], а u — часть подынтегрального выражения, которая будет дифференцироваться;
- integrand(expr) — выделение интегрируемой функции из выражения expr, содержащего интеграл отложенного действия.

Поясним некоторые из этих команд примерами.

Применим подстановку для вычисления интеграла:

```
> with(student): v:=Int(1/(3+2*cos(x)),x=0..Pi):
> v=expand(changevar(tan(x/2)=t,v,t));
```

$$\int_0^{\pi} \frac{1}{3+2\cos(x)}\,dx = 2\int_0^{\infty} \frac{1}{\left(1+\frac{4}{1+t^2}\right)(1+t^2)}\,dt$$

```
> value(v);
```

$$\frac{1}{5}\pi\sqrt{5}$$

Вычислим тройной интеграл:

```
> u:=Tripleint(x,z=0..sqrt((x+y)/2),
            y=0..x,x=0..2):%=value(%);
```

$$\int_0^2\int_0^x\int_0^{1/2\sqrt{2x+2y}} x\,dz\,dy\,dx = -\frac{32}{21}+\frac{64}{21}\sqrt{2}$$

Применим интегрирование по частям:

```
> Int(x*sin(x), x=0..Pi)=
  intparts(Int(x*sin(x), x=0..Pi), x); value(rhs(%));
```

$$\int_0^{\pi} x\sin(x)\,dx = \pi - \int_0^{\pi} -\cos(x)\,dx$$

$$\pi$$

В заключение описания возможностей интегрирования функций в Maple приведем несколько команд пакета student для приближенного интегрирования функции одной переменной на конечном отрезке и иллюстрации различных аппроксимаций интегралов:

- middlebox(f(x),x=a..b,n,options) — рисование графика функции f(x) вместе с аппроксимирующими интеграл прямоугольниками. Высота прямоугольника опре-

деляется значением функции `f(x)` в центре каждого интервала, параметр `n` задает число интервалов;

- `rightbox(f(x),x=a..b,n,options)` — рисование графика функции `f(x)` вместе с аппроксимирующими интеграл прямоугольниками. Высота прямоугольника определяется значением функции `f(x)` в правом конце каждого интервала, параметр `n` задает число интервалов;
- `middlesum(f(x), x=a..b, n)` — вычисление приближенного значения интеграла функции `f(x)` на отрезке `[a,b]`. Для аппроксимации используются прямоугольники с высотой, определяемой значением функции `f(x)` в центре каждого интервала, параметр `n` определяет число интервалов.

В качестве `options` в первых двух командах могут выступать параметры двумерной графики (см. главу 6 «Графика Maple»).

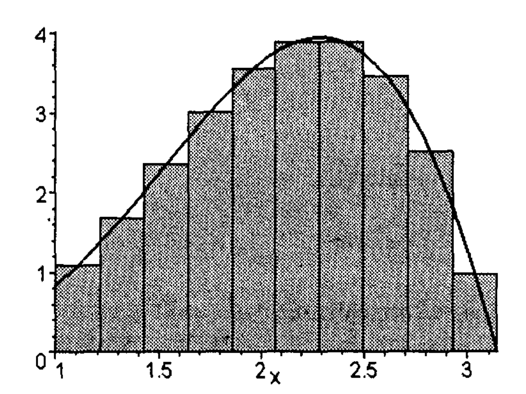
Пример:

```
> with(student): middlesum(x^2*sin(x), x=1..Pi, 10);
  evalf(%);
```

$$\left(\frac{1}{10}\pi-\frac{1}{10}\right)\left(\sum_{i=0}^{9}\left(1+\left(i+\frac{1}{2}\right)\left(\frac{1}{10}\pi-\frac{1}{10}\right)\right)^2\sin\left(1+\left(i+\frac{1}{2}\right)\left(\frac{1}{10}\pi-\frac{1}{10}\right)\right)\right)$$

5.669485513

```
> middlebox(x^2*sin(x), x=1..Pi, 10);
```



Интегральные преобразования

Ряд важных команд стандартной библиотеки предназначен для проведения интегральных преобразований Фурье, Лапласа, Меллина и др. Все эти команды находятся в пакете **inttrans**, и для его подключения нужно выполнить команду `with(inttrans)`.

Например, для вычисления преобразования Фурье применяется команда:

```
fourier(f,t,w)
```

Выражение `f` может включать экспоненты, полиномы, тригонометрические функции, дельта-функцию Дирака (`Dirac(t)`), функцию Хевисайда (`Heaviside(t)`), а также

производные и интегралы. Можно задавать также нестандартные преобразования Фурье. Параметр t определяет переменную трансформации, а w — параметр преобразования.

Далее перечислим в алфавитном порядке некоторые другие команды, которые выполняют соответствующее преобразование для выражения expr по переменной var и с новой переменной name:

- fouriercos(exp,var,name) — косинус-преобразование Фурье;
- fouriersin(exp,var,name) — синус-преобразование Фурье;
- invfourier(exp,var,name) — обратное преобразование Фурье;
- invlaplace(exp,var,name) — обратное преобразование Лапласа выражения expr по переменной var и с новой переменной name;
- laplace(exp,var,name) — преобразование Лапласа;
- mellin(exp,var,name) — преобразование Меллина.

Приведем пример обращения к процедурам прямого, обратного и косинус-преобразования Фурье:

```
> with(inttrans):
> fourier(1/(1+t), t, s);
```

$$I e^{(I s)} \pi (\text{Heaviside}(-s) - \text{Heaviside}(s))$$

```
> invfourier(%, s, t);
```

$$\frac{1}{1+t}$$

```
> fouriercos(1/(1+t), t, s);
```

$$\frac{\sqrt{2}\,(-\sin(s)\,\text{Ssi}(s) - \cos(s)\,\text{Ci}(s))}{\sqrt{\pi}}$$

Команды fourier и invfourier аналитически вычисляют прямое и обратное преобразования Фурье. Для анализа числовых данных можно применять команды Maple FFT и iFFT, реализующие алгоритмы прямого и обратного быстрых преобразований Фурье.

Естественно, описанные команды не исчерпывают всех возможностей Maple в области математического анализа. Большое число команд, и даже библиотек, написано многочисленными пользователями и распространяется по компьютерным сетям. Освоение этого материала требует затрат времени, и иногда оказывается проще написать свои программы, реализующие те или иные операции.



Решение уравнений в Maple

В данной главе рассмотрены средства пакета Maple 6, предназначенные для решения алгебраических уравнений и неравенств, обыкновенных дифференциальных уравнений и уравнений в частных производных.

С каждой версией и реализацией в Maple появляются новые возможности проведения преобразований, решения задач и, соответственно, добавляются новые команды и пакеты (библиотеки). Постоянно совершенствуются многоцелевые команды с корнем `solve`: решения алгебраических уравнений и неравенств `solve`, обыкновенных дифференциальных уравнений `dsolve` и уравнений в частных производных `pdsolve`. Кроме того, развиваются пакеты для решения разностных (`LREtools`) и дифференциальных уравнений (`DEtools`), а также уравнений в частных производных (`PDEtools`). Для представления решений, не записываемых явными формулами, введены специальные структуры: для алгебраических уравнений — `RootOf`, для разностных — `RESol` и для дифференциальных уравнений — `DESol`. Пакет Maple пополняется различными типами дифференциальных уравнений с известными аналитическими решениями, развивается аппарат преобразования уравнений в частных производных и совершенствуются методы численного анализа. Можно предположить дальнейшие развитие этих пакетов, и поэтому обсуждение их возможностей будет кратким.

Решение алгебраических уравнений и неравенств

Команда `solve` является многоцелевой и применяется для аналитического решения алгебраических уравнений и их систем, неравенств и функциональных уравнений, вычисления тождеств. Команда `fsolve` предназначена для получения численных решений. Для работы с разностными уравнениями имеется специальная команда `rsolve`.

Команда solve

Вызов команды solve имеет следующий вид:

```
solve (EQN,VAR)
```

Здесь EQN — уравнение или система уравнений, а VAR — переменная или группа переменных. Если параметр VAR отсутствует, то будут найдены решения относительно всех неизвестных, входящих в уравнения EQN. Отметим, что под неизвестными здесь понимаются все символьные переменные. Система уравнений и группа переменных задаются в виде множеств. Напомним, что совокупность объектов, разделенных запятыми и взятых в фигурные скобки, является множеством.

Уравнения могут быть заданы непосредственно в теле команды, а могут быть присвоены некоторой переменной. Если в качестве уравнения указано выражение без знака равенства, то считается, что это одна часть равенства, а другая равна нулю:

```
> sol:=solve(x^4-8*x,x);
```

$sol := 0, 2, -1 + I\sqrt{3}, -1 - I\sqrt{3}$

Для хранения решений удобно ввести переменную и обращаться к конкретному решению по индексу, например:

```
> sol[3];
```

$-1 + I\sqrt{3}$

Отметим, что результатом решения одного уравнения будет переменная типа exprseq (последовательность выражений). Если в качестве параметра указана переменная типа set (множество), то решение будет представлено в виде множества равенств, где в левой части стоит имя переменной, а справа — значение:

```
> sol:=solve({x^4-8*x=0},x);
```

$sol := \{x = 0\}, \{x = 2\}, \{x = -1 + I\sqrt{3}\}, \{x = -1 - I\sqrt{3}\}$

Ответ в виде множества удобен для подстановки нужного решения в выражения, зависящие от той переменной, относительно которой разыскивалось решение:

```
> subs(sol[-2],x^2);
```

$(-1 + I\sqrt{3})^2$

Чтобы присвоить найденные значения переменным на весь сеанс, применяется команда assign. Для отмены назначения применяется команда unassign или операция присваивания переменной ее имени, взятого в кавычки:

```
> s:=solve({3*x-y=sqrt(a), 5*x-2*y=1},{x,y});
```

$s := \{y = -3 + 5\sqrt{a}, x = -1 + 2\sqrt{a}\}$

```
> assign(s); x, y;
```

$-1 + 2\sqrt{a}, -3 + 5\sqrt{a}$

```
> unassign("x"); y:='y': x,y;
```

x, y

Поиск решения может потребовать значительных ресурсов памяти и времени. Для нелинейных уравнений может быть найдено несколько решений (но не обязательно все), а может оказаться, что ни одного решения не найдено. В последнем случае

Maple просто выведет приглашение ввода, ожидая новой команды. Отметим еще раз, что если не указаны переменные, относительно которых должно быть найдено решение, то ответ будет получен для всех переменных.

Если в выражении ответа появилась функция RootOf, это означает, что Maple либо не может выразить корни в радикалах, либо это требует дополнительных усилий:

```
> sx:=solve(x^4+3*x^3-8*x+3,{x});
```

$$sx := \{x = \mathrm{RootOf}(_Z^4 + 3\,_Z^3 - 8\,_Z + 3, index = 1)\},$$
$$\{x = \mathrm{RootOf}(_Z^4 + 3\,_Z^3 - 8\,_Z + 3, index = 2)\},$$
$$\{x = \mathrm{RootOf}(_Z^4 + 3\,_Z^3 - 8\,_Z + 3, index = 3)\},$$
$$\{x = \mathrm{RootOf}(_Z^4 + 3\,_Z^3 - 8\,_Z + 3, index = 4)\}$$

Само решение при этом выражается через корни аргумента функции, стоящей внутри RootOf. Его можно найти численно:

```
> evalf(sx[1]);
```

$$\{x = .4027975553\}$$

Команда allvalues позволяет представить решение, используя радикалы. Например, для первого решения sx получим:

```
> allvalues(sx[1]);
```

$$\{x = -\frac{3}{4} + \frac{1}{4}\sqrt{37} - \frac{1}{4}\sqrt{-10 + 2\sqrt{37}}\}$$

К аналогичному результату приводит использование команды convert с параметром radical:

```
> convert(sx[3],radical);
```

$$\{x = -\frac{3}{4} - \frac{1}{4}\sqrt{37} + \frac{1}{4}\sqrt{-10 - 2\sqrt{37}}\}$$

Режим вывода решений без использования конструкции RootOf можно задать, присвоив системной переменной _EnvExplicit значение true. В общем случае корни полиномов пятой степени и выше не выражаются в радикалах. В Maple 6 команда solve выписывает решения для частных случаев, когда такие решения имеются.

Конструкция RootOf(EQ(X)=0,X) применяется для обозначения любого корня уравнения EQ(X)=0. Это удобно для проведения выкладок. Расширенный формат команды позволяет указать тот корень из множества корней, который находится вблизи значения Z:

```
RootOf(EQ(X)=0,X,Z)
```

Эта команда позволяет выделять также комплексные корни, задавая комплексные значения для Z. Например:

```
> evalf(RootOf(x^4-256=0,x,3));
```

4.000000000

```
> evalf(RootOf(x^4-256=0,x,2*I));
```

$4.\,I$

Для системы уравнений каждое решение дается множеством, а при нескольких ответах — последовательностью множеств. Рассмотрим решение простой системы:

```
> eq:=(y+2*x)*y=5*y,x^2+y-4=0: sol:=solve({eq},{x,y});
```

$sol := \{x=2, y=0\}, \{x=-2, y=0\}, \{y=3, x=1\}, \{y=3, x=1\}$

Отметим удивительный результат: последние два решения совпадают, так как Maple нашел решения и проинформировал об их кратности. Для проверки полезно подставить полученное решение в исходное уравнение:

```
> eval(eq,sol[1]);
```

$0=0, 0=0$

Для проверки всех решений удобно пользоваться командой map:

```
> map(subs,[sol],{eq});
```

$[\{0=0\}, \{0=0\}, \{15=15, 0=0\}, \{15=15, 0=0\}]$

Так как решения подставляются в уравнения, оформленные как тип set (множество), и одинаковые элементы из множества удаляются, то результаты первых двух подстановок означают одновременное обнуление правых и левых частей уравнений.

Если неизвестных больше, чем уравнений, то решение будет содержать свободные параметры. Например, для системы трех линейных уравнений с четырьмя неизвестными без указания неизвестных Maple выберет одну переменную в качестве параметра и выведет решение следующего вида:

```
> eq:={w+u+4*v+8*t=2,v*3+w=0,v=3*w-5*t}: solve(eq);
```

$$\{u=u, w=\frac{2}{5}-\frac{1}{5}u, v=-\frac{2}{15}+\frac{1}{15}u, t=\frac{4}{15}-\frac{2}{15}u\}$$

Явное указание неизвестных позволяет получить решение с нужным параметром:

```
> solve(eq,{u,v,w});
```

$$\{u=2-\frac{15}{2}t, v=-\frac{1}{2}t, w=\frac{3}{2}t\}$$

Надо иметь в виду, что порядок следования переменных (и решений) может измениться при следующем запуске. Если для использования решений необходим определенный порядок следования неизвестных, то полезно упорядочить ответ при помощи команд сортировки или конструкций программирования, см. главу 16 «Программирование в Maple».

Для успешной работы могут понадобиться следующие переменные системы: при помощи _MaxSols указывается число разыскиваемых решений, для поиска всех решений нужно присвоить значение true переменной _EnvAllSolutions. При представлении решения используются следующие переменные: _Z (комплексное число), _NN (неотрицательное целое), _B (бинарное число — 0 или 1). Заданием переменной _EnvTryHard значения true может привести к решению, выраженному более компактным образом, но это потребует определенных затрат времени:

```
> _EnvAllSolutions:=true: r:=solve(tan(x)=1,x);
```

$$r := \frac{1}{4}\pi + \pi_Z1\sim$$

Для сравнения приведем решение с альтернативным значением:

```
> _EnvAllSolutions:=false: r:=solve(tan(x)=1,x);
```

$$r := \frac{1}{4}\pi$$

Заметим, что определение всех решений чревато затратами времени на анализ полученного результата. Потребуются некоторые усилия, чтобы отсечь неверные корни в ответе следующего примера:

```
> eq:=2*sin(x)^3-7*sin(x)^2+4*sin(x)+4: solve(eq,x);
```

$$-\frac{1}{6}\pi + \frac{4}{3}\pi_B1\sim + 2\pi_Z1\sim, \arcsin(2) - 2\arcsin(2)_B2\sim + 2\pi_Z2\sim + \pi_B2\sim$$

Если решить то же уравнение, взяв в качестве неизвестной синус, то ошибочные решения (синус больше единицы) видны сразу:

```
> solve(eq,{sin(x)});
```

$$\{\sin(x) = \frac{-1}{2}\}, \{\sin(x) = 2\}, \{\sin(x) = 2\}$$

В завершение отметим, что команда solve позволяет находить параметры, при которых справедливо некоторое тождество. Тождество IDEN задается при помощи команды identity(IDEN,VAR), причем тождество должно выполняться при любых значениях переменной VAR.

Приведем пример. Подготовим полином четвертой степени с неизвестными параметрами. Команда indets позволяет выделить множество неопределенных параметров в заданном выражении. После удаления переменной x получаем пять неизвестных параметров:

```
> eq:=sum(a[k]*x^k,k=0..4):var:=indets(eq) minus {x};
```

$$eq := a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

$$var := \{a_4, a_1, a_2, a_3, a_0\}$$

Теперь определим, при каких значениях будет справедливо следующее тождество:

```
> solve(identity(eq=(x-1)^3,x),var);
```

$$\{a_4 = 0, a_0 = -1, a_3 = 1, a_2 = -3, a_1 = 3\}$$

Команда fsolve

Для численного решения системы уравнений EQN относительно переменных VAR используется команда fsolve:

```
fsolve (EQN,VAR,OPT)
```

При помощи дополнительных параметров OPT могут быть заданы условия, устанавливающие местоположение, тип и число разыскиваемых решений (см. табл. 4.1), где x обозначает переменную, a, b, s — вещественные числа, n — целое число.

Для одного уравнения Maple старается найти вещественный корень, в случае полинома разыскиваются все вещественные корни. Приведем примеры решений и использования различных параметров.

Таблица 4.1. Параметры команды fsolve

Параметр	**Назначение**
a..b или x=a..b	Задание интервала [a,b] для поиска решений
avoid={x=s}	Поиск решений, отличных от x=s
complex	Поиск комплексных решений (одного для трансцендентных уравнений и всех для полиномов)
fulldigits	Использование арифметики с мантиссой, определенной константой Digits
maxsols=n	Поиск n наименьших решений (для полиномов)
x=s	Начальное приближение

Определим полином шестой степени и применим команду fsolve без параметров:

```
> poly:=x^6-2*x^5-4*x^4-6*x^3+15*x^2+56*x-60:
> fsolve(poly);
```

-2., 1., 2., 3.000000000

Найдем два наименьших корня полинома:

```
> fsolve(poly,x,maxsols=2);
```

-2., 1.

Найдем вещественные и комплексные корни, отличающиеся от двух указанных выше:

```
> fsolve(poly,x,complex,avoid={x=-2,x=3});
```

$-1. - 2.\,I, -1. + 2.\,I, 1., 2.$

При решении трансцендентных уравнений желательно указывать предполагаемое положение корней. В следующем примере без определения параметров будет найдено решение x=-1. Чтобы найти другие решения, используем два способа указания цели: задание близкого к корню начального приближения и определение диапазона:

```
> fsolve(cos(Pi*x)=x,x=.5);
```

.3769670094

```
> fsolve(cos(Pi*x)=x,x,-0.9..-0.5);
```

-.7898326284

Нужный диапазон или начальную точку можно определить, построив график функции (выражения с одной переменной), для которой разыскивается решение. В случае системы нелинейных уравнений это сделать труднее. Для систем с двумя и тремя неизвестными можно воспользоваться командами implicitplot и implicitplot3d, см. главу 6 «Графика Maple». Приведем пример решения системы, демонстрирующий, кстати, что при определении диапазона можно использовать бесконечность:

```
> fsolve({x^2+y^2=4,x^3=ln(y)},{x,y},{x=0..2,y=0..infinity});
```

$\{x = .8414345327, y = 1.814383622\}$

Решение неравенств

Команда solve позволяет находить решения одного неравенства относительно одной переменной. В ответе могут присутствовать следующие функции: RealRange

для указания интервала и Open для обозначения открытого интервала (граница не включена).

Определим неравенство:

```
> ineq:=2*ln(x^2-3)<=3-ln(x^2-3)^2;
```

$$ineq := 2\ln(x^2-3) \le 3 - \ln(x^2-3)^2$$

и запустим команду solve:

```
> solve(ineq,x);
```

$$\mathrm{RealRange}(-\sqrt{e+3}, -\sqrt{e^{(-3)}+3}), \mathrm{RealRange}(\sqrt{e^{(-3)}+3}, \sqrt{e+3})$$

Теперь превратим ineq в строгое неравенство и присвоим переменной so первое решение:

```
> so:=solve(lhs(ineq)<rhs(ineq))[1];
```

$$so := \mathrm{RealRange}(\mathrm{Open}(-\sqrt{e+3}), \mathrm{Open}(-\sqrt{e^{(-3)}+3}))$$

Используя команду op для доступа к операндам и обычный способ выбора элемента из последовательности выражений, можно добраться до значения нужной границы интервала:

```
> op(so); op(so)[1]; op(op(so)[1]);
```

$$\mathrm{Open}(-\sqrt{e+3}), \mathrm{Open}(-\sqrt{e^{(-3)}+3})$$

$$\mathrm{Open}(-\sqrt{e+3})$$

$$-\sqrt{e+3}$$

При помощи пакета Maple можно также решать смешанные системы, состоящие из нескольких уравнений и неравенства:

```
> solve({2*x+y>3,x^2=y,x*z=y},{x,y,z});
```

$$\{y=z^2, z<-3, x=z\}, \{y=z^2, 1<z, x=z\}$$

Заметим также, что и для линейных неравенств может быть получен некий ответ:

```
> solve({2*x+y>3,x>y},{x,y});
```

$$\{-2x-y<-3, y-x<0, 1<x\}$$

Для решения задач с несколькими неравенствами, например для задач линейного программирования, имеется пакет simplex, см. главу 8 «Математические библиотеки Maple».

Команды isolve и msolve

Кроме solve и fsolve имеется ряд специализированных команд: isolve(eqn) для отыскания решений уравнения eqn в целых числах и msolve(eqn,m) для нахождения решений уравнения eqn по модулю m.

Приведем для каждой команды по примеру, не требующему особых комментариев:

```
> isolve(m^2+2*n^2=257,{m,n});
```

$$\{m=15, n=4\}, \{m=15, n=-4\}, \{m=-15, n=4\}, \{m=-15, n=-4\}$$

```
> msolve(3^i=3,7);
```

$\{i = 1 + 6_Z1\sim\}$

Разностные уравнения

Для решения разностных уравнений или рекуррентных соотношений (линейные задачи с постоянными коэффициентами и некоторые нелинейные задачи) имеется команда

```
rsolve (EQN,VAR)
```

Здесь VAR — имя функции (набор имен функций), относительно которой (которых) будет решаться разностное уравнение (система уравнений) EQN. Если решение может быть получено, то ответ будет в виде функции от параметра. Помимо самих уравнений в EQN могут быть заданы начальные или граничные условия. При их отсутствии Maple постарается получить ответ в общем виде. Возможны следующие формы задания граничных условий:

```
V(n)=a, V(n..m)=a, V(k=n..m)=F(k)
```

Приведем пример решения линейного разностного уравнения второго порядка:

```
> eq:=3*f(n)=4*f(n-1)-f(n-2):
rsolve({eq,f(1)=0,f(2)=2},{f});
```

$$\{f(n) = 3 - 9\left(\frac{1}{3}\right)^n\}$$

С помощью параметра 'makeproc' команды rsolve можно определить процедуру. Как известно, полиномы Чебышева могут быть заданы следующей рекуррентной формулой:

```
> tt:=t(n+1)=2*x*t(n)-t(n-1):
T:=rsolve({tt,t(0)=1,t(1)=x},t,'makeproc');
```

```
T := proc(n)
local i, s, t, bipow;
    bipow := proc(n) ... end proc ;
    if 1 < nargs or not type(n, integer) then 'procname'(args)
    else

        s := bipow(n - 1);
        t := 0;
        for i to 2 do t := t + s[1, i]×(array(1 .. 2,[(1)=x,(2)=1]))[i] end do ;
        t
    end if

end proc
```

```
> T(3);
```

$(4x^2 - 1)x - 2x$

Отметим, что процедура была создана автоматически, программирование процедур рассматривается в главе 7 «Программирование в Maple».

Используем для проверки имеющуюся в Maple стандартную функцию ChebyshevT, определяющую полином Чебышева (первого рода):

```
> expand(T(3))=expand(ChebyshevT(3,x));
```

$$4x^3 - 3x = 4x^3 - 3x$$

Для квалифицированной работы с некоторыми типами линейных разностных уравнений создана библиотека **LREtools**. Для работы с ее командой **COM** может использоваться вызов с префиксом LREtools[COM] или должна быть подключена вся библиотека:

```
> with(LREtools);
```

[*REcontent, REcreate, REplot, REprimpart, REreduceorder, REtoDE, REtodelta, REtoproc, autodispersion, constcoeffsol*, δ, *dispersion, divconq, firstlin, hypergeomsols, polysols, ratpolysols, riccati, shift*]

Ограничимся тривиальным примером. Введем неавтономное разностное уравнение:

```
> lre:=g(n+1)=(19/10+1/(n+2))*g(n)-g(n-1);
```

$$lre := \mathrm{g}(n+1) = \left(\frac{19}{10} + \frac{1}{n+2}\right)\mathrm{g}(n) - \mathrm{g}(n-1)$$

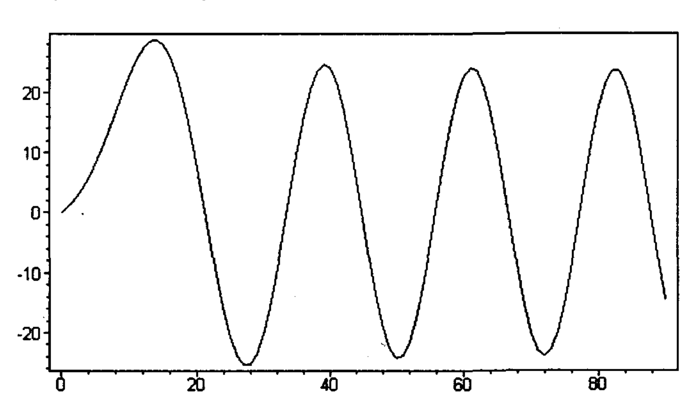
Обращение к rsolve приведет к повторению введенной команды:

```
> rsolve({lre,g(0)=0,g(1)=1},{g});
```

$$\mathrm{rsolve}\left(\left\{\mathrm{g}(n+1) = \left(\frac{19}{10} + \frac{1}{n+2}\right)\mathrm{g}(n) - \mathrm{g}(n-1), \mathrm{g}(0) = 0, \mathrm{g}(1) = 1\right\}, \{g\}\right)$$

Обратимся к процедуре графического построения решения:

```
> REplot(lre,g(n),{g(0)=0,g(1)=1},0..90,axes=boxed);
```



Для вычисления решения можно определить процедуру (текст не приводится):

```
> fun:=REtoproc(lre,g(n),{g(0)=0,g(1)=1}):
```

С ее помощью можно вычислить значение для нужной итерации (шага) n. Результат выводится рациональной дробью, поэтому рядом дадим десятичное представление:

```
> fun(7), evalf(fun(7),5);
```

$\frac{3454277273}{252000000}$, 13.707

Для обозначения решения разностного уравнения введена структура RESol — аналог структуры RootOf, представляющей решение разностного уравнения в том случае, когда явного решения нет.

Обыкновенные дифференциальные уравнения

В данном разделе речь пойдет о решении обыкновенных дифференциальных уравнений (ОДУ). Вначале рассмотрим применение команды dsolve для аналитического и численного решения ОДУ. Затем опишем специальную структуру DESol, позволяющую работать с неявно заданным решением дифференциального уравнения. Наконец, представим пакет DEtools, команды которого предназначены для преобразования дифференциальных уравнений и поиска точных решений, численного решения задачи Коши и визуализации результатов расчетов.

Для получения аналитических, приближенных и численных решений дифференциальных уравнений применяется команда dsolve, причем во всех случаях используется единый формат команды:

```
dsolve (ODE,VAR,OPT)
```

Здесь ODE – дифференциальное уравнение или система дифференциальных уравнений относительно неизвестных функций VAR. Для решения задачи Коши в уравнения ODE нужно включить также начальные условия, а для краевой задачи — соответственно краевые условия. Дополнительные условия OPT позволяют указать способ решения (type=...) и используемый метод (method=...). Например, для получения численного решения в качестве OPT задается type=numeric, степенные разложения будут строиться при type=series, а метод Рунге–Кутты–Фельберга применяется для численного интегрирования, если введена строка method=rkf45. При определении способа решения левая часть "type=" может быть опущена.

В уравнениях для указания производной применяются команды diff и оператор D, а для обозначения производной в начальных и краевых условиях используется оператор D. Система уравнений оформляется в виде множества: уравнения, начальные или краевые условия записываются через запятые и берутся в фигурные скобки.

Аналитические решения ОДУ

По умолчанию (дополнительные условия OPT не указаны), так как принято, что type=exact, Maple старается найти явное выражение для неизвестной функции (функций). При невозможности выделить искомую функцию (например, решение выражается через дробные степени) решение выводится в неявном виде. Если требуется решение в явном виде, то необходимо указать дополнительное условие

type=explicit (или просто explicit). Константа-параметр _EnvExplicit определяет способ представления решения.

Если число краевых или начальных условий меньше порядка системы, то в ответе будут фигурировать неопределенные константы _C1, _C2 и т. д. Для уравнений первого порядка может быть использовано параметрическое представление, и тогда в ответе будет фигурировать константа _T.

Рассмотрим пример задания дифференциального уравнения и обращения к команде dsolve. Обратим внимание, что при неполном задании уравнения (отсутствует знак равенства и правая часть) система Maple пополняет уравнение нулевой правой (или левой) частью:

```
> de:=diff(y(x),x$2)+4*diff(y(x),x)+3*y(x)+2;
```

$$de := \left(\frac{\partial^2}{\partial x^2}\,y(x)\right)+4\left(\frac{\partial}{\partial x}\,y(x)\right)+3\,y(x)+2$$

```
> dsolve(de,y(x));
```

$$y(x)=-\frac{2}{3}+_C1\,e^{(-3\,x)}+_C2\,e^{(-x)}$$

Решение линейного уравнения второго порядка найдено в виде суммы экспонент с произвольными константами. То же уравнение при помощи оператора D запишется в следующем виде:

```
> de:=(D@@2)(y)(x)+4*D(y)(x)+3*y(x)+2;
```

$$de := (D^{(2)})(y)(x)+4\,D(y)(x)+3\,y(x)+2$$

Рассмотрим задачу Коши для этого уравнения. Определим начальные условия и обратимся к команде dsolve:

```
> ic:=D(y)(0)=0,y(0)=-1; dsolve({de,ic},y(x));
```

$$ic := D(y)(0)=0,\ y(0)=-1$$

$$y(x)=-\frac{2}{3}-\frac{1}{2}\,e^{(-x)}+\frac{1}{6}\,e^{(-3\,x)}$$

Отметим, что переменная ic имеет тип exprseq (последовательность выражений), и объединение ее с уравнением de потребовало заключения их в фигурные скобки для оформления выражения типа set (множество). Если же переменную ic сразу определить как множество, то первый аргумент команды dsolve должен быть оформлен как сумма множеств. Для этого нужно взять переменную de в фигурные скобки и применить специальную операцию объединения (union). Найдем решение краевой задачи. Для этого зададим краевые условия в виде множества и обратимся к команде dsolve:

```
> bvp:={D(y)(0)=0,y(1)=-1}; dsolve({de} union bvp,y(x));
```

$$bvp := \{D(y)(0)=0,\ y(1)=-1\}$$

$$y(x)=-\frac{2}{3}-\frac{e^{(-x)}}{3\,e^{(-1)}-e^{(-3)}}+\frac{\frac{1}{3}\,e^{(-3\,x)}}{3\,e^{(-1)}-e^{(-3)}}$$

Команда dsolve предоставляет возможность определения базисных функций фундаментального решения дифференциального уравнения при указании параметра output= basis. Приведем соответствующий пример, заодно напомнив, что Maple легко обрабатывает символьные выражения, где бы они ни встретились, — здесь первый параметр команды содержит выражение, задающее новое, чуть измененное уравнение:

```
> dsolve(de-a*exp(x),y(x),output=basis);
```

$$\left[[e^{(-x)}, e^{(-3\,x)}], -\frac{2}{3}+\frac{1}{8}\,a\,e^{x}\right]$$

В Maple 6 можно просматривать ход решения при использовании dsolve. Для этого надо выполнить команду

```
> infolevel [dsolve]:=3;
```

$$infolevel_{dsolve} := 3$$

В результате для линейного дифференциального уравнения пятого порядка с постоянными коэффициентами имеем:

```
> de:=(D@@5)(y)(x)-(D@@3)(y)(x)+a*D(y)(x)-a*y(x); dsolve(de);
```

$$de := (D^{(5)})(y)(x) - (D^{(3)})(y)(x) + a\,D(y)(x) - a\,y(x)$$

```
Methods for high order ODEs:
Trying to isolate the derivative d^5y/dx^5...
Successful isolation of d^5y/dx^5
-> Trying classification methods
trying a quadrature
trying high order exact linear fully integrable
trying linear constant coefficient
linear constant coefficient successful
```

$$y(x) = _C2\,e^{x} + \left(\sum_{_a=1}^{4} _C1_{_a}\,e^{(\mathrm{RootOf}(_Z^4+_Z^3+a,\,index=_a)\,x)}\right)$$

Ответ содержит структуру RootOf, причем в качестве индекса выступает переменная _a. Функции RootOf появляется, если для дифференциального уравнения не удается факторизовать характеристический полином.

Для проверки того, что найденное решение SOL удовлетворяет уравнению или системе ODE, имеется функция odetest:

```
odetest (SOL,ODE)
```

Например, для рассмотренного линейного дифференциального уравнения пятого порядка с постоянными коэффициентами получим:

```
> odetest(%,de);
```

0

Если после подстановки решения в уравнение (систему уравнений) нуля не получилось, то можно попробовать преобразовать ответ, используя команды combine, expand и т. п.

Для решения дифференциальных уравнений и задач с начальными данными могут быть применены интегральные преобразования. Для этого в команде dsolve

следует указать дополнительный параметр method=TRANS, где в качестве TRANS взято одно из следующих слов, специфицирующих применяемое преобразование: laplace, fourier, fouriercos, fouriersin. Для использования метода laplace должен быть определен весь набор начальных условий.

Приведем простой пример:

```
> de:=diff(y(t),t,t)-y(t)=Heaviside(1-t);
```

$$de := \left(\frac{\partial^2}{\partial t^2}\mathrm{y}(t)\right) - \mathrm{y}(t) = \mathrm{Heaviside}(1-t)$$

```
> dsolve({de,D(y)(0)=0},y(t),method=fouriercos);
```

$$\mathrm{y}(t) = (1 - \mathrm{e}^{(-1)}\cosh(t) - \mathrm{e}^{(-t)}\sinh(1))\,\mathrm{Heaviside}(t-1) - 1 + \mathrm{e}^{(-1)}\cosh(t)$$

Отметим, что решение ОДУ или системы ОДУ может быть получено с учетом наложенных ограничений, которые задаются уравнениями или неравенствами. Например, найдем такое решение нелинейного уравнения второго порядка, что оно обращается в нуль при x=0, и существует точка a, в которой производная от решения также равняется нулю:

```
> ode:=y(x)*diff(y(x),x,x)+diff(y(x),x)+1=0;
```

$$ode := \mathrm{y}(x)\left(\frac{\partial^2}{\partial x^2}\mathrm{y}(x)\right) + \left(\frac{\partial}{\partial x}\mathrm{y}(x)\right) + 1 = 0$$

```
> simplify(dsolve({ode,y(0)=0,diff(y(a),a)=0},y(x)));
```

$$\mathrm{y}(x) = -\frac{\left(-1 + \ln\left(-\frac{-x + x\,\mathrm{e} - \mathrm{e}\,a}{a}\right)\right)(-x + x\,\mathrm{e} - \mathrm{e}\,a)}{-1 + \mathrm{e}}$$

Для упрощения полученного решения сразу применили команду simplify.

Приближенные решения ОДУ

Для решения нелинейных задач, которые не поддаются аналитическим методам, могут быть использованы численные процедуры. При этом, однако, должны быть определены численные значения всех параметров. Когда решения в замкнутой форме нет, а численные подходы невозможны или нежелательны, то могут быть использованы приближенные методы. Для этого команда dsolve имеет параметр series для поиска решения в виде разложения в ряд и параметр laplace для применения метода преобразования Лапласа.

Приведем примеры обращения к команде dsolve с параметром series. Вначале переопределим значение переменной Order, задающей порядок для вычисления разложений, а затем получим решение для нелинейного уравнения второго порядка:

```
> Order:=4: de:=(D@@2)(y)(x)-y(x)^2=sin(x);
> dsolve(de,y(x),series);
```

$$de := \left(\frac{\partial^2}{\partial x^2}\mathrm{y}(x)\right) - \mathrm{y}(x)^2 = \sin(x)$$

$$\mathrm{y}(x) = \mathrm{y}(0) + \mathrm{D}(y)(0)\,x + \frac{1}{2}\mathrm{y}(0)^2 x^2 + \left(\frac{1}{3}\mathrm{y}(0)\,\mathrm{D}(y)(0) + \frac{1}{6}\right)x^3 + \mathrm{O}(x^4)$$

В ответе фигурируют неопределенные начальные значения для функции и первой производной. При помощи команды subs легко задать нужные значения:

```
> subs(y(0)=0,D(y)(0)=1,%);
```

$$y(x) = x + \frac{1}{6}x^3 + O(x^4)$$

Рассмотрим еще один пример:

```
> restart:Order:=6: de:=D(y)(x)-sin(b*y(x))=x:
> sol:=dsolve({de,y(0)=0}, y(x),series);
```

$$de := D(y)(x) - \sin(b\,y(x)) = x$$

$$sol := y(x) = \frac{1}{2}x^2 + \frac{1}{6}b\,x^3 + \frac{1}{24}b^2x^4 + \frac{1}{120}b^3x^5 + O(x^6)$$

Выполним проверку. Для этого используем команду odetest и разложение в ряд, чтобы избавиться от синуса.

```
> te:=series( map(odetest,[sol],de)[] ,x);
```

$$te := O(x^5)$$

Для получения решений линейных дифференциальных уравнений в виде формальных степенных рядов в Maple создан пакет Slode, предназначенный для нахождения полиномиальных, дробно-рациональных, гипергеометрических и других разложений.

Численные решения ОДУ

Для численного решения дифференциальных уравнений при помощи команды dsolve нужно указать параметр numeric. Это можно сделать двумя способами:

```
dsolve(ODE,VAR,type=numeric,OPT)
```

или

```
dsolve (ODE,VAR,numeric,OPT)
```

Уравнения и начальные условия задаются в виде множества ODE, VAR есть неизвестная функция или множество (список) неизвестных функций, дополнительные параметры OPT определяют метод численного решения задачи Коши и ряд параметров метода. В качестве правой части конструкции method=MET могут выступать: метод Рунге–Кутты–Фельберга порядка 4/5 (rkf45), метод Рунге–Кутты порядка 7/8 (dverk78), одношаговый (gear) и многошаговый (mgear) методы Гира и набор методов для решения систем жестких уравнений lsode. Ряд классических методов можно определить при помощи конструкции method= classical[MET], здесь в качестве MET можно указать различные варианты метода Эйлера (foreuler, heunform, impoly), несколько методов Рунге–Кутты (rk2, rk3, rk4), многошаговые методы (adambash, admoulton). Дополнительно могут быть определены метод разложения в ряд Тейлора для вычисления решения (taylorseries) и метод интегральных преобразований (inttrans). Для ряда методов имеется своя система параметров, описание которых можно получить, обратившись к справке. По умолчанию для интегрирования используется метод Рунге–Кутты–Фельберга.

По умолчанию в результате выполнения команды dsolve с параметром numeric будет создана процедура, к которой можно обращаться для вычисления отдельных значений и для построения графика решения на заданном промежутке.

Приведем пример решения задачи Коши для нелинейной системы двух уравнений первого порядка, описывающей колебания физического маятника с линейным трением:

```
> des:=D(x)(t)=y(t),D(y)(t)=-a*y(t)-.8*sin(x(t));
> ic:=x(0)=0,y(0)=2;
```

$des := D(x)(t) = y(t), D(y)(t) = -.2\ y(t) - .8\sin(x(t))$

$i\tilde{n} := x(0) = 0, y(0) = 2$

```
> F:=dsolve({des,ic},{x(t),y(t)},numeric);
```

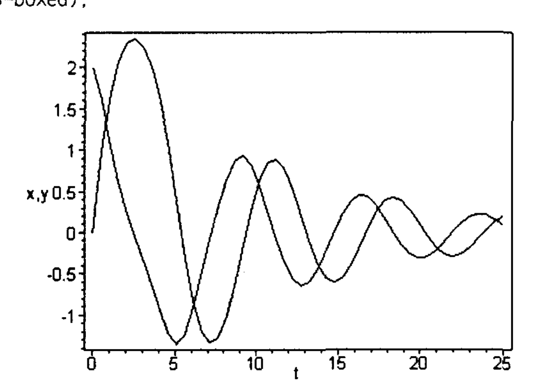
F := **proc** (*rkf45_x*) ... **end proc**

```
> a:=.2; evalf(F(1.5),5);
```

$a := .2$

$[t = 1.5, x(t) = 2.0540, y(t) = .65462]$

```
> plots[odeplot](F,[[t,x(t)],[t,y(t)]],0..25,
labels=[t,"x,y"],axes=boxed);
```



В приведенном примере для построения графика решения использована команда odeplot из пакета plots. Для визуализации численных решений задачи Коши удобнее использовать графические команды пакета DEtools, которые рассмотрены ниже и обеспечивают различные средства для отображения результатов интегрирования дифференциальных уравнений.

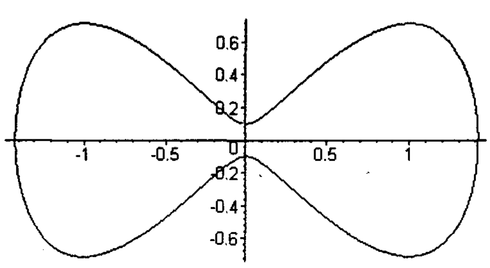
При помощи параметра output=listprocedure сформируется список процедур — по процедуре на каждую переменную:

```
> p:=dsolve({diff(x(t),t)=y(t),diff(y(t),t)=-x(t)^3+x(t),
x(0)=0,y(0)=0.1},{x(t),y(t)},
type=numeric,output=listprocedure);
```

p := [*t* = (**proc** (*t*) ... **end proc**), x(*t*) = (**proc** (*t*) ... **end proc**),
y(*t*) = (**proc** (*t*) ... **end proc**)]

Далее к решению можно обращаться, указывая нужную переменную. Для того чтобы получить фазовый портрет решения, используется параметрическое задание графика для команды plot:

```
> X,Y:=subs(p,x(t)),subs(p,y(t)): plot([X,Y,0..16]);
```



Чтобы при помощи команды dsolve получить решение в виде таблицы, можно использовать параметр value=X с массивом точек X, для которых будет вычислено решение:

```
> ode:={D(y)(t)=y(t)*(1-y(t)),y(0)=.5}:
arr:=array([0,.5,1.0,2.0,5.0]):
F:=dsolve(ode,y(t),type=numeric,method=gear,value=arr);
```

$$F := \begin{bmatrix} [t, y(t)] \\ \begin{bmatrix} 0 & .5 \\ .5 & .6224593304 \\ 1.0 & .7310585770 \\ 2.0 & .8807970769 \\ 5.0 & .9933071488 \end{bmatrix} \end{bmatrix}$$

Для каждого метода интегрирования имеется свой набор дополнительных параметров точности и др. Например, для метода Рунге–Кутты–Фельберга можно задать величину абсолютной погрешности abserr=VAL (по умолчанию VAL=Float(1,2-Digits)), минимум относительной погрешности minrel=VAL (по умолчанию aerr= Float(1,1-Digits)) и относительную погрешность relerr=VAL (VAL=Float(1,2-Digits)). Здесь первым параметром команды Float задается мантисса, а вторым — порядок формируемого числа. Дополнительно можно указать предельные количества для вычисления правой части (maxfun) и выводимых точек (maxkop), ввод отрицательных значений для этих параметров задает расчет без ограничений.

Можно написать собственную процедуру интегрирования, например My_Scheme. Для использования этой процедуры нужно указать в перечне параметров method= My_Scheme. Заголовок процедуры при этом должен иметь следующий вид:

```
My_Scheme:=proc(var,h,t_t,iter,pt,F),
```

где var — число переменных, h — шаг интегрирования, t_t — длина отрезка интегрирования, iter — число шагов между точками, запасаемыми в массиве pt для вывода на график (каждая точка дается списком [t,x,y]), F — функция, вычисляющая правые части системы дифференциальных уравнений.

Структура DESol

Даже в том случае, когда отсутствует явное представление решения, можно работать с решением дифференциального уравнения или системы. Для этого вводится специальный объект DESol:

```
DESol (ODE,VAR,INI)
```

По данной команде для уравнения или системы ODE относительно неизвестной функции или функций VAR с начальными данными INI создается специальная структура. Это своеобразное развитие функции RootOf применительно к дифференциальным уравнениям. Объект DESol можно интегрировать, дифференцировать, находить для него разложения и, конечно, получать численное решение.

Например, используя DESol, можно проверить, является ли некоторое выражение интегралом данного уравнения. Определим объект DESol в качестве решения уравнения физического маятника:

```
> sol:=DESol(diff(y(x),x,x)+a*sin(y(x)),y(x));
```

$$sol := \mathrm{DESol}\left(\left\{\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) + a\sin(\mathrm{y}(x))\right\}, \{\mathrm{y}(x)\}\right)$$

Запишем выражение предполагаемого интеграла и проверим, равна ли нулю производная интеграла в силу системы:

```
> integ:=diff(sol,x)^2/2-a*cos(sol): diff(integ,x);
```

$$0$$

Следующим примером покажем, как использовать объект DESol для получения приближенного решения разложением в ряд (команда series) и построения численного решения. Зададим уравнение Дуффинга и начальные условия:

```
> de:=diff(y(x),x,x)+y(x)+y(x)^3=cos(x);
> ic:={y(0)=0,D(y)(0)=1};
```

$$de := \left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) + \mathrm{y}(x) + \mathrm{y}(x)^3 = \cos(x)$$

$$ic := \{\mathrm{y}(0) = 0, \mathrm{D}(y)(0) = 1\}$$

Определим сам объект DESol:

```
> des:=DESol(de,y(x),ic);
```

$$des := \mathrm{DESol}\left(\left\{\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) + \mathrm{y}(x) + \mathrm{y}(x)^3 - \cos(x)\right\}, \{\mathrm{y}(x)\}, \{\mathrm{y}(0) = 0, \mathrm{D}(y)(0) = 1\}\right)$$

При попытке аналитического решения задачи Коши для данного уравнения область вывода останется пустой:

```
> dsolve({de} union ic, y(x));
```

Применим команду series для вычисления решения в виде ряда по степеням x, а полученное выражение преобразуем в полином при помощи команды convert:

```
> Poly:=convert(series(des,x=0),polynom);
```

$$Poly := x + \frac{1}{2}x^2 - \frac{1}{6}x^3 - \frac{1}{12}x^4 - \frac{1}{24}x^5$$

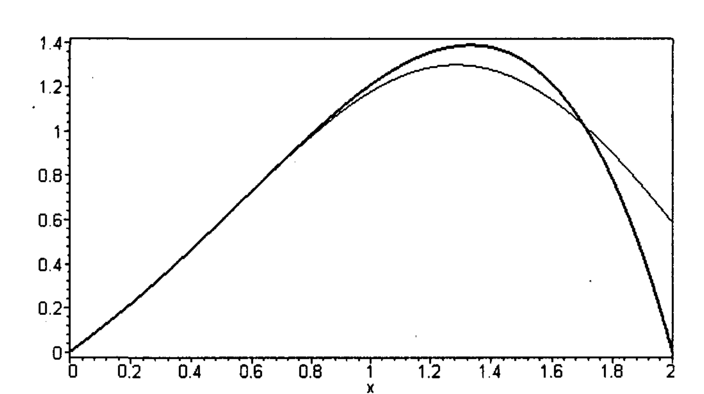
Затем сгенерируем процедуру численного решения:

```
> Y:=dsolve({de} union ic, y(x),type=numeric);
```

Y := proc (*rkf45_x*) ... end proc

Теперь по команде odeplot из графического пакета **plots** построим график численного решения, при помощи универсальной команды plot — график полинома, и полученные рисунки объединим командой display:

```
> pY:=plots[odeplot](Y,[x,y(x)],0..2,color=black):
> pP:=plot(Poly,x=0..2,thickness=2,color=black):
> plots[display]({pY,pP},axes=boxed);
```



На графике более тонкая кривая отвечает численному решению исходной задачи.

Пакет DEtools

Для квалифицированной работы с обыкновенными дифференциальными уравнениями, будь то определение типа ОДУ, преобразование ОДУ к иному виду, использование аппарата теории симметрий Ли, численное решение задачи Коши или построение фазовых портретов и векторных полей, предназначен пакет **DEtools**.

Для использования команд из пакета **DEtools** можно подключить весь пакет командой

```
with(DEtools)
```

или загрузить нужную команду COM при помощи обращения

```
with(DEtools,COM)
```

Кроме того, можно использовать полную форму обращения, указывая в квадратных скобках имя команды после имени пакета:

```
DEtools[COM](…)
```

В настоящее время пакет состоит из нескольких групп команд.

Подпакет **Poincare** недавно включен в Maple. В его состав входят команда получения гамильтоновой системы уравнений по заданному гамильтониану (hamilton_eqs), команда подготовки начальных точек для интегрирования generate_ic, главная команда poincare для получения двумерных или трехмерных сечений Пуанкаре и команда zoom, позволяющая увеличивать или уменьшать масштаб изображения без перерасчета траекторий.

Для преобразования дифференциальных уравнений имеется следующая группа команд: autonomous, convertAlg, convertsys, DEnormal, indicialeq, reduceOrder, regularsp,

translate, untranslate, varparam. Так, команда convertAlg пытается понизить порядок рассматриваемого уравнения, а команда regularsp вычисляет равновесия (сингулярные точки) для линейных уравнений. Команда convertAlg применяется для получения коэффициентов дифференциального уравнения, а команда DEnormal — для нормализации его коэффициентов, причем может применяться как к уравнению, так и к списку коэффициентов. Поясним действие этих команд простым примером:

```
> de:=D(y)(x)/x+diff(y(x),x,x)+x; DE:=convertAlg(de,y(x));
```

$$de := \frac{D(y)(x)}{x} + \left(\frac{\partial^2}{\partial x^2} y(x)\right) + x$$

$$DE := \left[\left[0, \frac{1}{x}, 1\right], -x\right]$$

```
> DEnormal(de,x,y(x));
```

$$\left(\frac{\partial}{\partial x} y(x)\right) + \left(\frac{\partial^2}{\partial x^2} y(x)\right) x = -x^2$$

```
> DEnormal(DE,x);
```

$$[[0, 1, x], -x^2]$$

Для преобразования уравнения высокого порядка к системе уравнений первого порядка служит команда, работу которой демонстрирует следующий пример:

```
> convertsys({de},{y(0)=a},{y(x)},x,y,d_y);
```

$$\left[\left[d_y_1 = y_2, d_y_2 = -\frac{y_2 + x^2}{x}\right], \left[y_1 = y(x), y_2 = \frac{\partial}{\partial x} y(x)\right], 0, [a, D(y)(0)]\right]$$

Ответ дается в виде списка, состоящего, в свою очередь, из списка уравнений, где в левой части уравнений использовано обозначение производных (в примере d_y), списка замен, начальной точки и списка начальных значений.

Команды, используемые для преобразования дифференциальных уравнений, могут применяться к уравнениям в частных производных, и наоборот. Например, команда dchange из пакета **PDEtools** позволяет проводить замену переменных в дифференциальных уравнениях. В предыдущих версиях для этого в пакете **DEtools** имелась команда Dchangevar.

Большая группа команд предназначена для классификации ОДУ, а также для преобразования ОДУ с использованием интегрирующих множителей и групп симметрий Ли (см. справку ?DEtools,Lie). Не вдаваясь в пояснения, ограничимся перечнем имен: Xchange, Xcommutator, Xgauge, buildsol, buildsym, canoni, convert_ODEs, equinv, eta_k, firint, firtest, gensys, infgen, intfactor, invariants, line_int, muchange, mutest, normalG2, odeadvisor, odepde, redode, reduce_order, remove_RootOf, solve_group, symgen, symtest, transinv.

Важную роль для обучения и поиска решений играет команда odeadvisor, которая представляет информацию о типе рассматриваемого уравнения или системы. Например, для уравнения второго порядка определяется его тип — это уравнение Бесселя:

```
> ode:= x^2*(D@@2)(y)(x)+x*D(y)(x)-(x^2+1)*y(x)=0;
```

$$ode := x^2 (D^{(2)})(y)(x) + x\, D(y)(x) - (x^2 + 1)\, y(x) = 0$$

```
> DEtools[odeadvisor](ode);
```

[[_*Bessel*, _*modified*]]

```
> dsolve(ode);
```

$y(x) = _C1\ \mathrm{BesselI}(1, x) + _C2\ \mathrm{BesselK}(1, x)$

Для работы с линейными дифференциальными уравнениями и построения формальных разложений имеются команды пакета Slode и специальная структура LODEstruct. Можно также свести исходное дифференциальное уравнение к уравнению меньшего порядка, решения которого являются также решениями исходной задачи, и работать с дифференциальными операторами, для деталей см. справку ?diffop. Для преобразования уравнения к дифференциальному оператору используется команда de2diffop, а для обратного преобразования — diffop2de.

Следующая группа команд обеспечивает построение решений в замкнутой форме: Dfactorsols, RiemannPsols, abelsol, bernoullisol, chinisol, clairautsol, constcoeffsols, eulersols, exactsol, expsols, genhomosol, kovacicsols, liesol, linearsol, matrixDE, parametricsol, polysols, ratsols, riccatisol, separablesol.

Разработчики Maple стремятся к тому, чтобы все уравнения из справочников по ОДУ были представлены в пакете. Из-за невозможности даже короткого описания типов точно решаемых уравнений ограничимся беглым перечислением команд, указывая корневое слово, если имеется несколько видов уравнений (и команд) данного класса.

Команды анализа уравнения первого порядка включают следующие типы: Абеля (Abel), однородные (homogeneous), Бернулли (Bernoulli), Риккати (Riccati) и др. (Chini, Clairaut, dAlembert), а также линейные (linear) и точно решаемые уравнения (exact), вычисляемые в квадратурах (quadrature), допускающие разделение переменных (separable), с дробно-рациональной правой частью (rational) и т. д.

Приведем пример решения линейного дифференциального уравнения с переменными коэффициентами:

```
> lin_ode:=diff(y(x),x)+sin(x)*y(x)-sin(2*x);
```

$$lin_ode := \left(\frac{\partial}{\partial x} y(x)\right) + \sin(x)\, y(x) - \sin(2\, x)$$

```
> DEtools[odeadvisor](lin_ode); so:=dsolve(lin_ode);
```

[_*linear*]

$$so := y(x) = 2\cos(x) + 2 + e^{\cos(x)}\, _C1$$

Отметим, что анализу и решению поддаются уравнения с произвольными функциями в качестве коэффициентов:

```
> lde:=diff(y(x),x)+f(x)*y(x)-g(x);
```

$$lde := \left(\frac{\partial}{\partial x} y(x)\right) + f(x)\, y(x) - g(x)$$

```
> DEtools[odeadvisor](lde); so:=dsolve(lde);
```

[_*linear*]

$$so := y(x) = \left(\int g(x)\, e^{\left(\int f(x)\, dx\right)} dx + _C1\right) e^{\left(\int -f(x)\, dx\right)}$$

Для решения уравнений второго порядка имеются команды, ориентированные как на отдельные важные уравнения, так и на целые классы уравнений: Bessel, Duffing, ellipsoidal, elliptic, Emden, erf, Gegenbauer, Halm, Hermite, Jacobi, Lagerstrom, Laguerre, Lienard, Liouville, linear_sym, Painleve, quadrature, sym_Fx, Titchmarsh, Van_der_Pol.

Для решения уравнений второго и высших порядков применяются следующие команды: exact_linear, exact_nonlinear, linear_ODEs, missing, quadrature, reducible.

Рассмотрим в качестве примера уравнение Ван дер Поля [МЭ]:

```
> VdP:=diff(y(x),x,x)-mu*(1-y(x)^2)*diff(y(x),x)+y(x);
DEtools[odeadvisor](VdP); ans:=dsolve(VdP,way=3);
```

$$VdP := \left(\frac{\partial^2}{\partial x^2}\,y(x)\right) - \mu\,(1 - y(x)^2)\left(\frac{\partial}{\partial x}\,y(x)\right) + y(x)$$

$$[[_2nd_order, _missing_x], _Van_der_Pol]$$

$$ans := y(x) = _a \;\&\text{where}\left[\ \{\left(\frac{\partial}{\partial _a}\,_b(_a)\right)_b(_a) - \mu\,_b(_a) + \mu\,_b(_a)\,_a^2 + _a = 0\},\right.$$

$$\left.\{_a = y(x),\ _b(_a) = \frac{\partial}{\partial x}\,y(x)\},\ \left\{x = \int \frac{1}{_b(_a)}\,d_a + _C1,\ y(x) = _a\right\}\right]$$

Ответ представлен специальной структурой ODESolStruc, которая появляется в том случае, если для рассматриваемого дифференциального уравнения был понижен порядок, но решение не было получено. Структура содержит термин &where и состоит из двух полей: функционального представления в новых переменных и списка трех множеств (редуцированное дифференциальное уравнение, прямое и обратное преобразования переменных).

Конечно, простое перечисление типов дифференциальных уравнений мало информативно и для использования упомянутых команд требуется обращение к справочной системе. В то же время любое изложение темы ОДУ будет неполным уже в момент написания вследствие усилий разработчиков Maple, затрачиваемых на развитие разделов, посвященных решению дифференциальных задач.

Графические команды пакета DEtools

Команды визуализации численных решений задачи Коши для дифференциальных уравнений и систем приведены в табл. 4.2.

Таблица 4.2. Команды визуализации пакета DEtools

Имя	Назначение
DEplot	Двумерные графики решений дифференциального уравнения или системы уравнений
Deplot3d	Трехмерные графики решений дифференциального уравнения или системы уравнений
dfieldplot	Изображение двумерного поля направлений (векторного поля)
phaseportrait	Фазовый портрет для системы уравнений первого порядка

Начнем описание графических возможностей пакета DEtools с команды DEplot, предназначенной для вывода графиков решений системы дифференциальных уравнений. Возможные форматы обращения к команде имеют вид:

```
DEplot(ODE,VAR,T_T,INI,OPT)
```

и

```
DEplot(ODE,VAR,T_T,INI,X_X,Y_Y,OPT)
```

Здесь приняты следующие обозначения для параметров: ODE — система n уравнений первого порядка или одно уравнение n-го порядка; VAR — имена переменных; T_T — область изменения независимой переменной; INI — начальные условия; X_X и Y_Y — масштабы изменения функций решения; OPT — дополнительные параметры. Хотя для проведения расчета требуется задать много параметров (метод, шаг, точность и т. д.), разработчики Maple постарались выбрать разумные назначения для стандартных ситуаций. Поэтому дополнительные параметры OPT могут отсутствовать.

При интегрировании уравнений все символические переменные, кроме имен функций и независимой переменной, должны получить численные значения. Первый аргумент ODE может принимать разные формы. Например, для системы из двух уравнений первого порядка допустимы следующие варианты:

```
[diff(x(t),t)=f1(t,x,y), diff(y(t),t)=f2(t,x,y)]
```

или

```
[D(x)(t)=f1(t,x,y), D(y)(t)=f2(t,x,y)]
```

Через f1(t,x,y), f2(t,x,y) обозначены правые части дифференциальных уравнений, в которых могут присутствовать независимая переменная t, искомые функции x,y, а также константы. В случае уравнения высокого порядка уравнения должны быть разрешены относительно старшей производной:

```
[diff(y(t),t$n)=f(t,y,...)],
```

Здесь n — целое число, а знак $ используется для указания производной порядка n>1, функция f(t,x) может включать также производные от переменной y до n-1 порядка включительно.

Параметром VAR даются имена переменных. Возможны две формы задания VAR: [x,y,...] или [x(t),y(t),...].

Параметр T_T определяет интервал изменения независимой переменной и может задаваться в виде a..b или t=a..b, причем a и b должны быть вещественными константами. Диапазоны вывода для неизвестных функций X_X и Y_Y определяются аналогично.

Параметр INI задает список начальных условий, где каждый набор оформляется в виде списка. Даже для единственного набора начальных условий должен быть подготовлен список списков:

```
[ [x(t0)=x0,y(t0)=y0], [x(t1)=x1,y(t1)=y1],...]
```

Видно, что стартовые значения независимой переменной (t) для разных начальных точек могут не совпадать, а решение для неизвестных функций будет определено для всего интервала T_T, так что задание начальных значений INI специфици-

рует различные интегральные кривые. Если число начальных условий меньше числа переменных, то выводится сообщение и график не строится.

Для уравнения n-го порядка относительно функции y(t) вначале задается значение независимой переменной, а далее значения функции и производных: [t0,y0,y'0,y''0,...].

Информацию обо всех параметрах команды DEplot можно получить, обратившись к справке:

```
?DEtools[options]
```

По умолчанию для неавтономных уравнений строятся кривые решений в двумерном пространстве первых двух переменных VAR. Для получения иных проекций или графиков зависимости различных функций от независимой переменной нужно задать параметры сцены (scene=...). Для системы двух автономных уравнений фазовый портрет для исходных переменных строится по умолчанию.

В графических командах пакета **DEtools** можно использовать параметры команд пакета **plots**. Так, для задания заголовка можно применять конструкцию title=STR, здесь **STR** — строка. Описание предоставляемых при этом возможностей и примеры будут даны в главе 6 «Графика Maple». Специфичные для пакета **DEtools** параметры представлены в табл. 4.3, где использованы следующие обозначения: BOOL — булева константа, N, M — целые числа, VAL — вещественное число, TYP — термин, MET — имя метода численного решения задачи Коши, в скобках даны принятые по умолчанию назначения.

Таблица 4.3. Параметры пакета DEtools

Имя	Назначение
x=c1..d1, y=c2..d2 или x(t)=c1..d1, y(t)=c2..d2	Границы вывода для переменных x и y указываются при помощи вещественных констант
arraysize=N	Размер рабочего массива для результатов интегрирования
arrows=TYP	Тип стрелок при изображении поля направлений, здесь TYP — один из следующих терминов: NONE (по умолчанию), THIN, SLIM, THICK, LINE
grid=[N,M]	Сетка для вывода поля направлений (grid=[10,10])
iterations=N	Число шагов интегрирования между выводимыми точками (iterations=1). Этот параметр полезен для повышения точности расчета, поскольку запоминаются те точки, которые выводятся на график
limitrange=BOOL	Прекращение расчета, если интегральная кривая вышла за диапазон, определенный для переменной x или y. По умолчанию limitrange=false. Если диапазоны для x и y не определены, то расчет ведется независимо от данного параметра
method=MET	Задание метода интегрирования
stepsize=VAL	Шаг интегрирования, по умолчанию VAL=(b-a)/20, где [a,b] — отрезок интегрирования

Поскольку Maple не является специализированной программой для решения систем дифференциальных уравнений, то нужно иметь в виду, что уменьше-

ние шага дает не только повышение точности и гладкости получаемых кривых, но и увеличение времени расчета. Так как все выводимые на график точки запасаются в памяти, то возможно ее исчерпание. По умолчанию для интегрирования применяется метод Рунге–Кутта, что эквивалентно заданию method=classical[rk4].

Параметр scene задает вид рисунка. Пусть задана система дифференциальных уравнений с неизвестными функциями x(t), y(t), ... Тогда scene=[x,y] означает изображение двумерного фазового портрета (график функции y(x)), а scene=[t,x,y] и scene=[x,y,t] означают трехмерные картины с различным расположением осей. Для одного уравнения высокого порядка выводится только график решения.

В табл. 4.3 в качестве MET выступает название одного из методов, перечень допустимых методов дан выше в разделе, описывающем применение команды dsolve для численного интегрирования задачи Коши.

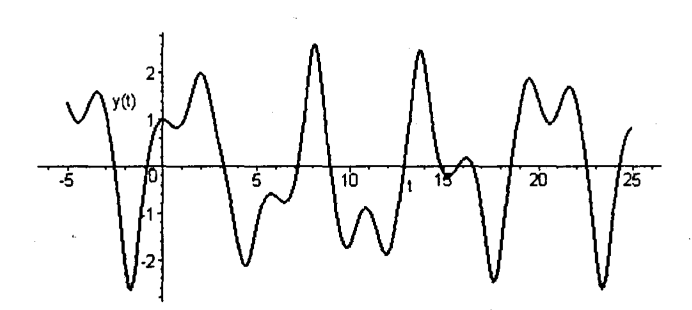
Приведем примеры использования команды DEplot для визуализации решения уравнения Дуффинга с вынуждающей гармонической силой. Подготовим процедуру для вычисления правой части системы дифференциальных уравнений:

```
> ode:=proc(n,t,y,yp)
   yp[1]:=y[2]; yp[2]:=-y[1]*(1+y[1]^2)+5*sin(t)
end proc;
```

$$ode := \mathbf{proc}\,(n, t, y, yp)\; yp[1] := y[2];\; yp[2] := -y[1]\times(1 + y[1]\verb|^|2) + 5\times\sin(t)\;\mathbf{end\ proc}$$

Поскольку в качестве неизвестной функции для решаемого неавтономного уравнения указана y(t), то по умолчанию строится траектория в пространстве (t,y). Обратим внимание, что число неизвестных указывается при помощи параметра number=N до начальных условий:

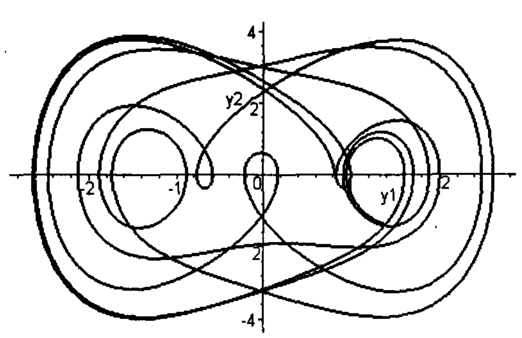
```
> DEplot(ode,y(t),t=-5..25,number=2,[[y(0)=1,D(y)(0)=0]],
thickness=2,stepsize=.05,arrows=none,linecolor=black);
```



Отметим, что график y(t) построен для всего интервала изменения независимой переменной t ([-5..25]), хотя начальные условия заданы в точке t=0.

Назначив в качестве неизвестных список из двух функций, получим фазовый портрет:

```
> DEplot(ode,[y1,y2],t=-5..25,number=2,[[y1(0)=1,y2(0)=0]],
stepsize=.05,arrows=none,linecolor=black,thickness=2);
```



Команда Deplot3d имеет тот же формат, что и рассмотренная команда DEplot, но с поправкой на заданную размерность решаемых уравнений:

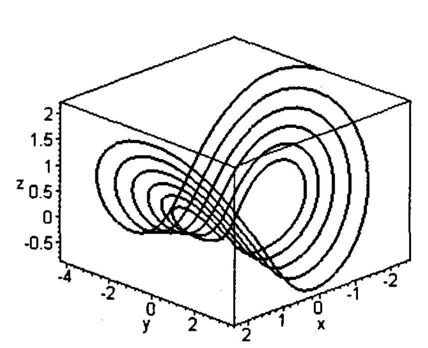
```
> restart:
> de3:=diff(x(t),t)=y(t),diff(z(t),t)=-nu*z(t)+x(t)*y(t),
diff(y(t),t)=-mu*y(t)-(z(t)-1)*x(t)-x(t)^3;
```

de3 :=

$$\frac{\partial}{\partial t}\,\mathrm{x}(t)=\mathrm{y}(t),\ \frac{\partial}{\partial t}\,\mathrm{z}(t)=-\nu\,\mathrm{z}(t)+\mathrm{x}(t)\,\mathrm{y}(t),\ \frac{\partial}{\partial t}\,\mathrm{y}(t)=-\mu\,\mathrm{y}(t)-(\mathrm{z}(t)-1)\,\mathrm{x}(t)-\mathrm{x}(t)^3$$

Определим параметры:

```
> mu:=-.1: nu:=.1:
> DEtools[DEplot3d]([de3],[x,y,z],t=0..25,stepsize=.05,
[[x(0)=1,y(0)=1,z(0)=0]],orientation=[45,60],
arrows=none,linecolor=black,thickness=2);
```



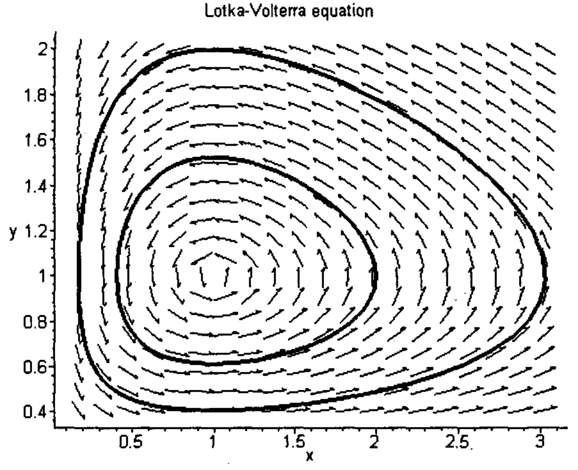
Команда phaseportrait предназначена для изображения фазовых кривых на плоскости и обращение к ней аналогично рассмотренному для команды DEplot:

```
phaseportrait(ODE,VAR,T_T,INI,OPT)
```

Здесь по-прежнему ODE — дифференциальное уравнение или система уравнений первого порядка, VAR — имена переменных в квадратных скобках, T_T — интервал изменения независимой переменной, INI — множество начальных условий, OPT — дополнительные параметры, частично описанные выше для команды DEplot. Для трехмерных картин поле направлений не строится.

Приведем пример построения фазовых кривых для системы уравнений Лотки–Вольтерра:

```
> with(DEtools):
f:=[D(x)(t)=3*x(t)*(1-y(t)),D(y)(t)=y(t)*(x(t)-1)]:
ini:=[[x(0)=2,y(0)=1],[x(0)=1,y(0)=2]]:
phaseportrait(f,[x(t),y(t)],t=0..7,ini,linecolor=black,
color=black,stepsize=.05,title="Lotka-Volterra equation");
```



Поскольку в качестве разделителя взято двоеточие, то область вывода для задающей дифференциальное уравнение команды отсутствует. Здесь и далее в простых случаях команды с опущенным выводом используются для сокращения информации, дублируемой в областях ввода и вывода.

Команда dfieldplot предназначена для изображения векторного поля (поля направлений) и согласована по параметрам с командой DEplot, за исключением начальных условий INI, задания которых не требуется. Можно сказать, что для двумерных задач команда DEplot есть объединение команд dfieldplot и phaseportrait. Примеры использования команд пакета содержатся также в главе 10 «Примеры решения задач».

Уравнения в частных производных

Для решения одного уравнения в частных производных имееются команда pdsolve и набор команд пакета **PDEtools**. Обращение к команде pdsolve имеет следующий вид:

```
pdsolve (PDE,FUN,OPT)
```

Здесь PDE — уравнение, FUN — неизвестная функция, а OPT — дополнительные параметры. Пакет Maple пытается найти общее решение, используя набор специальных методов для некоторых видов уравнений и метод разделения переменных в общем случае. Ответ может быть представлен специальной структурой **PDESolStruc**, которая содержит термин &where и состоит из двух полей: функционального представления и перечня обыкновенных дифференциальных уравнений, полученных в результате процедуры разделения переменных. Данная структура предназначена для указания возможных решений. Решение при этом может быть получено с помощью команды build из пакета **PDEtools**.

В качестве дополнительных параметров OPT могут использоваться: указание на возможную форму решения HINT=HI, параметр INTEGRATE, означающий автоматическое интегрирование получающейся системы обыкновенных дифференциальных уравнений, указание на поиск явного решения build. В качестве HI могут стоять знаки суммы «+» и произведения «*» или алгебраическое выражение.

Чтобы проиллюстрировать работу команды pdsolve, рассмотрим линейное гиперболическое уравнение первого порядка:

```
> pd:=diff(u(x,t),t)+2/3*diff(u(x,t),x)=0;
```

$$pd := \left(\frac{\partial}{\partial t}\mathrm{u}(x,t)\right)+\frac{2}{3}\left(\frac{\partial}{\partial x}\mathrm{u}(x,t)\right)=0$$

В результате обращения к команде pdsolve получается общее решение «бегущая волна»:

```
> sol:=pdsolve(pd);
```

$$sol := \mathrm{u}(x,t)=_F1\left(t-\frac{3}{2}x\right)$$

При помощи команды piecewise организуем начальный профиль — полусинусоиду — и подставим его в решение, попутно образовав функцию для последующего построения графика:

```
> f1:=x->piecewise(x> 0 and x<Pi,2*sin(x));
```

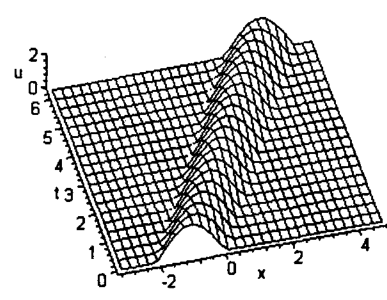
$$f1 := x \to \mathrm{piecewise}(0<x \text{ and } x<\pi, 2\sin(x))$$

```
> S:=unapply(rhs(eval(sol,_F1=f1)),x,t);
```

$$S := (x,t) \to \mathrm{piecewise}\left(-t+\frac{3}{2}x<0 \text{ and } t-\frac{3}{2}x-\pi<0, -2\sin\left(-t+\frac{3}{2}x\right)\right)$$

Для изображения полученного решения обратимся к команде plot3d:

```
> plot3d(S(x,t),x=-Pi..3/2*Pi,t=0..2*Pi,grid=[30,20],
style=hidden,axes=frame,orientation=[-105,10],
labels=["x","t","u"],color=black);
```



В качестве следующего примера рассмотрим уравнение теплопроводности:

```
> heat:=diff(u(x,t),t)=diff(u(x,t),x,x);
```

$$heat := \frac{\partial}{\partial t}\mathrm{u}(x,t)=\frac{\partial^2}{\partial x^2}\mathrm{u}(x,t)$$

Вначале будем разыскивать решение в виде произведения:

```
> pdsolve(heat,HINT="*");
```

$$(u(x,t) = _F1(x)\,_F2(t))\ \&where\ \left[\{\frac{\partial}{\partial t}_F2(t) = _c_1\,_F2(t), \frac{\partial^2}{\partial x^2}_F1(x) = _c_1\,_F1(x)\}\right]$$

Результат содержит неизвестные функции $_F_1(t)$ и $_F_2(t)$, а также константу $_c_1$. Попробуем задать явное представление для решения через функции X(x), T(t) и дополнительно укажем, чтобы решение было выписано явно:

```
> sol:=pdsolve(heat,HINT=X(x)*T(t),build);
```

$$sol := u(x,t) = e^{(\sqrt{_c_1}\,x)}\,_C1\,e^{(_c_1\,t)}\,_C2 + \frac{_C1\,e^{(_c_1\,t)}\,_C3}{e^{(\sqrt{_c_1}\,x)}}$$

Теперь определим параметры и получим частное решение:

```
> sub:={_c[1]=-4,_C1=I/2,_C2=-1,_C3=1}:
> S:=simplify(eval(rhs(sol),sub));
```

$$S := e^{(-4\,t)}\sin(2\,x)$$

Рассмотренный пример показывает, как, используя подсказку, можно находить решения командой pdsolve.

Пакет PDEtools

Пакет PDEtools предназначен для преобразования уравнений в частных производных и поиска аналитических решений, кроме того, в его состав входит графическая команда PDEplot. В табл. 4.4 представлены краткие описания команд пакета для преобразования уравнений, а возможности команды PDEplot рассмотрены далее.

Для работы с дифференциальными уравнениями, как обыкновенными, так и в частных производных, применяются различные преобразования, аппарат теории групп Ли и др. С развитием пакета Maple происходит унификация команд для проведения выкладок и нахождения решений близких по используемому аппарату дифференциальных задач. Так, команда dchange из пакета PDEtools позволяет проводить замену переменных в дифференциальных уравнениях и уравнениях в частных производных, интегралах и интегро-дифференциальных уравнениях, а также замещать глобальные переменные в процедурах. Обращение к команде dchange имеет вид:

```
dchange(TR,EXPR,NEW,ITR,known,unknown,param,proc)
```

Обязательными параметрами являются TR — множество соотношений, определяющих замену переменных, и EXPR — алгебраическое выражение или процедура, задающая уравнение. При необходимости указываются: список имен новых переменных NEW (если не очевидны новые переменные), ITR —множество соотношений, определяющих обратную замену. Дополнительными параметрами определяются имена известных функций known, имена функций, которые будут преобразованы, unknown, список имен констант params. Задание этих параметров (known,

unknown, params) производится по единой форме, например known=ИМЯ. Здесь ИМЯ — имя или множество имен. Последний параметр proc определяет процедуру, которая будет использоваться для упрощения полученного выражения (уравнения, интеграла).

Таблица 4.4. Команды пакета PDEtools

Имя	Назначение
build	Решение обыкновенных дифференциальных уравнений, полученных применением процедуры разделения переменных
casesplit	Представление дифференциальных уравнений в виде подсистем
charstrip	Получение уравнений характеристик для уравнения в частных производных
dchange	Замена переменных в уравнениях, кратных интегралах, интегро-дифференциальных уравнениях и процедурах
dcoeffs	Выделение коэффициентов при неизвестной и ее производных для полиномиального уравнения
difforder	Определение порядка частных производных
dsubs	Подстановка выражений для производных
mapde	Преобразование уравнений к другому виду
pdetest	Подстановка в уравнение предполагаемого решения и вычисление невязки
separability	Определение условий, при которых возможно решение уравнения методом разделения переменных
splitsys	Разложение системы уравнений на независимые подсистемы
splitstrip	Вычисление характеристик

Для иллюстрации работы команды dchange приведем замену переменных для уравнения гармонического осциллятора:

```
> with(PDEtools,dchange): de:=diff(x(t),t,t)*a^2+x(t)=0;
```

$[dchange]$

$$de := \left(\frac{\partial^2}{\partial t^2}\,\mathrm{x}(t)\right)a^2 + \mathrm{x}(t) = 0$$

```
> dchange({t=tau*a,x(t)=y(tau)},de,'params'=a);
```

$$\left(\frac{\partial^2}{\partial \tau^2}\,\mathrm{y}(\tau)\right) + \mathrm{y}(\tau) = 0$$

В следующем примере команда dchange применяется для уравнения в частных производных:

```
> pde:=diff(f(x,y),x)+diff(f(x,y),y)+g(x-y,y)*u(x)+b*v(x+y)=0;
```

$$pde := \left(\frac{\partial}{\partial x}\,\mathrm{f}(x, y)\right) + \left(\frac{\partial}{\partial y}\,\mathrm{f}(x, y)\right) + \mathrm{g}(x - y, y)\,\mathrm{u}(x) + b\,\mathrm{v}(x + y) = 0$$

```
> tr:={x=r+s,y=r-s}:dchange(tr,pde,known=g,uknown={u,v});
```

$$\left(\frac{\partial}{\partial r}\,\mathrm{f}(r, s)\right) + \mathrm{g}(2\,s, r - s)\,\mathrm{u}(r, s) + b\,\mathrm{v}(r) = 0$$

Команда PDEplot

Для изображения решения уравнения в частных производных первого порядка применяется команда PDEplot, обращение к ней имеет вид:

```
PDEplot (PDE,INI,DIA,OPT)
```

Здесь PDE определяет линейное или нелинейное уравнение относительно функции, зависящей от n переменных, начальные условия INI даются списком из n+1 элемента и таким образом специфицируют кривую в пространстве размерности n+1, параметр DIA задает диапазоны для выводимых переменных. При помощи параметра OPT формулируются дополнительные параметры.

Для вычисления решения (гиперповерхности, проходящей через кривую, определенную параметром INI) используется метод характеристик, и система обыкновенных дифференциальных уравнений интегрируется по умолчанию методом Рунге–Кутты четвертого порядка (classical[rk4]). Для решения задачи можно определить другой метод (см. описание методов в разделе, посвященном команде dsolve), однако это может замедлить процесс получения решения.

Таблица 4.5. Параметры команды PDEplot

Имя	Назначение
animate=BOOL	Анимация многообразий решения (animate=true). По умолчанию при n=2 действует animate=true и animate=false при n>2
basechar=BOOL	Режим отображения начального условия на плоскость (x,y) при basechar=true. По умолчанию принято basechar=false
color=COL	Цвет решения
ic_assumptions	Заданные в виде равенств или неравенств ограничения на начальные условия для первых производных
initcolor=COL	Цвет кривой начального условия
iterations=N	Число шагов интегрирования между вычисляемыми точками. По умолчанию iterations=1
method=MET	Численный метод интегрирования вдоль характеристик, см. перечень методов для команды dsolve
numchar=N	Число точек для построения гиперповерхности решения. Задается в виде списка или числа, по умолчанию numchar=20
numsteps=[N,M]	Количество вычисляемых точек. По умолчанию numsteps=[-10,10]
obsrange=BOOL	Прекращение (obsrange=true) интегрирования, если отображаемая переменная вышла из заданного диапазона, и режим безусловного расчета при obsrange=false
scene=[x,y,u(x,y)]	Определение осей рисунка
stepsize=VAL	Шаг интегрирования, по умолчанию задано максимальное значение stepsize=0.25
xi=ximin..ximax, u(x,y,...)=umin..umax	Определение диапазонов отображения переменных и функций

В табл. 4.5 представлен перечень параметров команды PDEplot и использованы следующие обозначения: BOOL — булева константа, N, M — целые числа, VAL — веществен-

ное число, MET — имя метода численного решения задачи Коши, COL — цвет. Помимо перечисленных можно также использовать ряд параметров команд plot3d и dsolve. Перед вставкой параметра полезно посмотреть примеры из справки Maple или даже скопировать подходящий пример, чтобы переделать его, а не вводить команду целиком.

Рассмотрим применение команды PDEplot для изображения ударной волны — решения нелинейного волнового уравнения (уравнения Бюргерса):

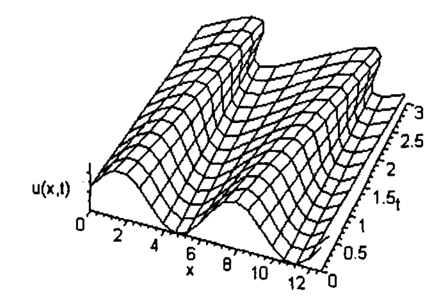
```
> pde:=diff(u(x,t),t)+u(x,t)*diff(u(x,t),x);
ini:=[s,0,.5+.5*sin(s)]; dia:=0..4*Pi;
```

$$pde := \left(\frac{\partial}{\partial t}\mathrm{u}(x,t)\right) + \mathrm{u}(x,t)\left(\frac{\partial}{\partial x}\mathrm{u}(x,t)\right)$$

$$ini := [s, 0, .5 + .5\sin(s)]$$

$$dia := 0\,..\,4\,\pi$$

```
> PDEtools[PDEplot](pde,x=dia,t=0..3,ini,s=dia,axes=frame,
style=hidden,orientation=[-70,15],color=black,numsteps=12);
```



5 ГЛАВА Алгебра в Maple

После описания возможностей пакета Maple для решения задач математического анализа и решения уравнений естественно перейти к рассмотрению другого классического математического курса – алгебры. Мы не ставим своей целью рассказать здесь обо всех имеющихся в Maple командах алгебры, а представим довольно подробный обзор основных и наиболее часто применяемых с нашей точки зрения. Основное внимание в этой главе будет уделено описанию команд линейной алгебры, однако в Maple существует возможность проведения выкладок и в операторном виде. Можно задавать свойства операторов и операции с ними, кроме того, есть пакет tensor для тензорных вычислений, пакет теории групп group и пакет Ore_algebra, реализующий основные конструкции алгебры линейных операторов.

Линейная алгебра

Небольшая часть команд для решения задач линейной алгебры содержится в стандартной библиотеке и пакете student, большая же часть находится в пакетах linalg и LinearAlgebra.

В состав предыдущих версий Maple входил только пакет linalg, который состоит более чем из сотни команд, реализующих основные операции линейной алгебры. Несмотря на эффективность реализованных в нем алгоритмов, этот пакет обладает некоторыми недостатками: для выполнения матричных операций нужно использовать или команду evalm, или специальные команды, а самое главное – ограниченные возможности работы с числовыми матрицами большого порядка. В пакете LinearAlgebra эти недостатки устранены и, кроме того, добавлены команды, основанные на высокоэффективных алгоритмах компании NAG (Numerical Algorithms Group). При реализации пакета LinearAlgebra использовались новые возможности языка Maple – модули (см. главу 7 «Программирование в Maple»).

Возможности пакетов во многом перекрываются, хотя можно сказать, что пакет linalg удобен для проведения выкладок линейной алгебры в символьном виде, а Linear-Algebra – для вычислений. Однако в каждом из пакетов существуют возможности,

не имеющие аналогов в другом пакете. В качестве демонстрационных примеров в этой главе используются упражнения из задачников Фаддеева и Соминского [69], Моденова и Пархоменко [64].

Напомним, что команды линейной алгебры становятся доступными после подключения пакетов linalg или LinearAlgebra соответственно командами

```
with(linalg)
```

и

```
with(LinearAlgebra)
```

Существуют и короткие способы вызова любой команды comm с указанием имени пакета:

```
linalg[comm]
```

и

```
LinearAlgebra[comm]
```

Кроме того, к команде comm из пакета LinearAlgebra можно обращаться как к элементу модуля:

```
LinearAlgebra:-comm
```

В Maple реализованы практически все операции линейной алгебры. Ниже приводится описание Maple-команд для большой части возможных операций. Более подробную информацию о командах пакетов linalg и LinearAlgebra можно получить, обратившись к справке: ?linalg и ?LinearAlgebra.

Матрицы и векторы

Основными объектами команд линейной алгебры являются матрицы и векторы. Пакеты linalg и LinearAlgebra оперируют с объектами различных типов, то есть представление матриц и векторов у этих пакетов различны. В этом разделе мы сначала опишем, как определяются матрицы и векторы в пакете linalg, а затем — в LinearAlgebra.

Вектором в linalg считается одномерный массив, который можно определить при помощи описателя array(1..n,[val1, val2,...]), где n — размерность вектора, а val1, val2,... — значения элементов. Другой способ описания вектора использует команду vector из пакета linalg:

```
vector (n,[val1,val2,...])
```

Здесь n — размерность вектора, а val1, val2, ... — значения элементов, причем если опустить параметр размерности, то размерность вектора определяется числом его элементов. Существует также другой вариант этого описателя: vector(n,f); здесь f — функция от индекса, используемая для генерации вектора.

В пакете linalg матрицей считается двумерный массив, у которого индексы изменяются от единицы, так что, например, переменная, описанная как array(0..3,1..4), матрицей не является. В этом пакете можно использовать два способа задания матриц: при помощи команд-описателей matrix и array. Коротко формат описателя array уже обсуждался выше, дополнительно отметим, что при помощи этой коман-

ды можно задавать матрицы специального вида. Параметры symmetric и antisymmetric указываются для определения симметричных и кососимметричных матриц, а для определения единичных и диагональных матриц используются соответственно параметры identity и diagonal.

Команда-описатель matrix из пакета linalg имеет вид:

```
matrix (n,m,[val1,val2,...])
```

здесь n — число строк, m — число столбцов матрицы, а val1, val2, ... — значения элементов матрицы. Синонимом данной команды является описание вида:

```
array(1..n,1..m, [val1,val2,...])
```

Вместе с тем существуют и другие формы описателя matrix, например matrix(n,m,f), здесь f — функция от двух целых переменных (индексов матрицы), с помощью которой присваиваются значения элементам матрицы.

Значения элементов матриц и векторов можно задавать как при описании, так и в ходе работы при помощи оператора присваивания, при этом часть элементов массива (матрицы, вектора) можно не определять. В последнем случае Maple заполнит вакансии переменными по умолчанию. Напомним, что для просмотра содержимого массивов нужно использовать команду eval или op.

При иллюстрации действия команд в этой главе мы будем считать, что результаты ранее приводимых примеров сохраняются.

Приведем несколько примеров задания матриц и векторов:

```
> A:=matrix(4,3,[[a,b,c],[],[1,2,3],[x,y,z]]);
```

$$A := \begin{bmatrix} a & b & c \\ A_{2,1} & A_{2,2} & A_{2,3} \\ 1 & 2 & 3 \\ x & y & z \end{bmatrix}$$

Элементы второй строки не были определены, и Maple заполнил их переменными по умолчанию.

```
> fun:=(i,j)->x^i/y^j: B:=matrix(2,3,fun);
```

$$B := \begin{bmatrix} \frac{x}{y} & \frac{x}{y^2} & \frac{x}{y^3} \\ \frac{x^2}{y} & \frac{x^2}{y^2} & \frac{x^2}{y^3} \end{bmatrix}$$

```
> u:=array(1..3); v:=vector([a,b,c]);
```

$u := \text{array}(1 .. 3, [\ ])$

$v := [a, b, c]$

```
> F:=array(1..3,1..3,antisymmetric): eval(F);
```

$$\begin{bmatrix} 0 & ?_{1,2} & ?_{1,3} \\ -?_{1,2} & 0 & ?_{2,3} \\ -?_{1,3} & -?_{2,3} & 0 \end{bmatrix}$$

```
> F[1,2]:=a: op(F);
```

$$\begin{bmatrix} 0 & a & ?_{1,3} \\ -a & 0 & ?_{2,3} \\ -?_{1,3} & -?_{2,3} & 0 \end{bmatrix}$$

В последнем примере знак вопроса означает неопределенность элементов матрицы.

В пакете linalg существуют также команды, позволяющие формировать матрицы специального вида. Так, команда hilbert(n,expr) генерирует гильбертову матрицу размерности n*n для выражения expr. Для задания блочной матрицы используется команда blockmatrix, случайная матрица задается командой randmatrix.

В пакете LinearAlgebra матрицы задаются с помощью команды Matrix (напомним, что в Maple большие и маленькие буквы различаются), а векторы — Vector. Существует короткий способ задания переменных типа Vector и Matrix при помощи символов >, < и |. Символы < и > определяют строки, а | — их разделитель. Пример:

```
> <<m,o>|<n,p>>; whattype(%);
```

$$\begin{bmatrix} m & n \\ o & p \end{bmatrix}$$

Matrix

Однако такое задание не всегда удобно и не предоставляет возможности управлять структурой объектов. Полные варианты команд задания векторов и матриц позволяют детально определять различные характеристики объектов, но довольно громоздки, хотя все параметры не являются обязательными.

Полный вариант команды Matrix имеет вид:

```
Matrix(r, c, init, ro, sc, sh, st, os, dt, f, a)
```

Здесь r — указывает число строк матрицы, а c — число столбцов. В качестве параметра init, который определяет значения элементов матрицы, могут фигурировать массивы, процедуры (аналогично команде matrix пакета linalg), переменные типа list и другие конструкции языка Maple. С помощью параметра ro можно запрещать изменение элементов матрицы, в этом случае он имеет вид: readonly=true. Параметр sc может применяться только в случае, если в качестве init фигурирует лист данных, и предназначен для управления заполнением матрицы (scan=rows — массив данных определяет строки, а scan=columns — столбцы матрицы). Для формирования матриц специального вида используются управляющие параметры sh и st. Например, для задания кососимметричной матрицы в качестве sh нужно указать shape=antisymmetric, а если вместо st поставить storage=diagonal, то будет сформирована диагональная матрица. Управлять способом хранения матрицы в памяти можно при помощи параметра os, если он имеет вид order=Fortran_order, то матрица хранится по столбцам, а при order=C_order — по строкам. Параметр dt указывает тип числовых данных (complex[8] — комплексные, integer[n] — целые, n=1, ..., 8). Задав параметр f в виде fill=value, можно определить значение всех элементов матрицы величиной value с типом, заданным параметром dt. И наконец, пара-

метр a, имеющий вид attributes=list, определяет дополнительные характеристики матрицы (подробно об этих характеристиках см. в справке Maple).

Вектор в пакете LinearAlgebra определяется при помощи команды

```
Vector[t](d, init, ro, sh, st, dt, f, a, t)
```

Здесь [t] указывает ориентацию вектора (row — вектор-строка, column — вектор-столбец), d — размерность вектора. Последний параметр t совпадает по содержанию с параметром [t], но отличается по написанию: orientation=name, где в качестве name фигурирует row или column. Назначения других параметров аналогичны одноименным параметрам команды Matrix.

Отметим еще раз то, что все параметры команд Matrix и Vector не являются обязательными и существуют правила определения свойств матриц и векторов по умолчанию. Например, если опущено указание числа столбцов, то матрица считается квадратной размерности r на r.

В отличие от пакета linalg, где элементы матриц и векторов принимают значение переменных по умолчанию в случае их неопределенности, в пакете LinearAlgebra незаданные элементы обнуляются. Если для объектов были определены свойства, а затем проделываются операции, этим свойствам противоречащие, то Maple выводит сообщение об ошибке. На экран переменные типа Matrix и Vector выводятся в виде матриц и векторов только для малых размерностей, а в других случаях выводится только информация об объекте. Отметим, что для просмотра переменных этих типов не нужно пользоваться командой eval, а достаточно к ним обратиться по имени.

Теперь приведем примеры задания матриц и векторов в пакете LinearAlgebra:

```
> Matrix(2);
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> Vector[row](3,[a,2,3]);
```

$$[a, 2, 3]$$

```
> Matrix([[1,2],[3,4]],scan=columns);
```

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

```
> Matrix([[1,2],[3,4]],scan=rows);
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> A1:=Matrix(1..3,3,shape=antisymmetric):
```

Обращение к элементам матрицы производится указанием индексов в квадратных скобках. При попытке задать значение, противоречащее свойству матрицы, следует сообщение об ошибке:

```
> A1[1,1]:=1;
Error, attempt to assign non-zero to antisymmetric diagonal
```

```
> A1[2,1]:=a-c: A1;
```

$$\begin{bmatrix} 0 & -a+c & 0 \\ a-c & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> Vector(425);
```

$$\begin{bmatrix} \text{425 Element Column Vector} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran_order} \end{bmatrix}$$

Отметим, что в качестве элементов матриц и векторов в пакете LinearAlgebra могут указываться переменные любого скалярного типа.

Для преобразования матричных и векторных переменных из типов пакета linalg в типы, с которыми работает пакет LinearAlgebra, используется команда convert с указанием соответствующего типа. Такая необходимость возникает при использовании команд пакета LinearAlgebra для работы с переменными, описанными как array. Отметим, что команды пакета linalg работают без проблем с переменными Vector и Matrix, однако результат их работы будет иметь тип array. Приведем пример, в котором используется команда вычисления определителя пакета LinearAlgebra. При первом обращении Maple выводит сообщение об ошибке, а после преобразования типа получается правильный результат:

```
> AA:=array(1..3,1..3,[[1,2,3],[4,5,6],[7,8,0]]):
  LinearAlgebra[Determinant](AA);
Error, LinearAlgebra:-Determinant expects its 1st argument, A, to be of type Matrix, but received AA
> LinearAlgebra[Determinant](convert(AA,Matrix));
```

27

При вычислении определителя объекта типа Matrix командой пакета linalg ошибки не возникает:

```
> AA:=Matrix([[a,2,3],[4,5,b],[7,8,0]]):linalg[det](AA);
```

$-8\,a\,b-9+14\,b$

Понятно, что объекты линейной алгебры (матрицы, векторы) могут быть получены и как результат преобразования массивов, списков, таблиц и т. д. Например, команда convert(list,vector) преобразует переменную типа list в вектор пакета linalg, а для преобразования набора переменных типа list в матрицу пакета LinearAlgebra используется команда convert([list1,...,listn],Matrix).

В пакете LinearAlgebra существуют специальные команды считывания (ImportMatrix) и записи матриц (ExportMatrix) на диск, которые описаны в разделе «Команды ввода/вывода» главы 7 «Программирование в Maple». Вывести графическое изображение структуры матрицы можно при помощи команды matrixplot (см. главу 6 «Графика Maple»).

Работа со структурой матрицы и вектора

Перечислим некоторые команды, позволяющие получать информацию о матричных и векторных переменных, корректировать эти объекты и работать с отдельными строками и столбцами.

Для выяснения размерностей матрицы M в пакете linalg имеются две команды: rowdim(M) — выводит число строк матрицы M, а coldim(M) — число столбцов. Например, для определенной выше матрицы A получим:

```
> rowdim(A); coldim(A);
```

4

3

Число элементов вектора v пакета linalg можно узнать при помощи команды vectdim(v).

Команды пакета LinearAlgebra для определения размерностей имеют вид: Dimension(M) — размерность матрицы или вектора, RowDimension(M) — число строк, ColumnDimension(M) — число столбцов. Пример:

```
> Dimension(convert(A,Matrix)); Dimension(Vector(345));
```

4, 3

345

Далее перечислим команды, позволяющие изменять размерности матриц, добавлять или удалять строки и столбцы. Для того чтобы удалить из матрицы (matrix) M строки с номерами от i до j, нужно воспользоваться командой delcols(M,i..j), а для удаления столбцов применяется команда delrows(M,i..j). Приведем пример удаления второй строки из матрицы A:

```
> C:=delrows(A,2..2);
```

$$C := \begin{bmatrix} a & b & c \\ 1 & 2 & 3 \\ x & y & z \end{bmatrix}$$

Для удаления строк и столбцов матрицы типа Matrix существуют аналогичные команды DeleteRow(A,L,outopts) и DeleteColumn(A,L,outopts). Здесь список L определяет номера удаляемых элементов, а outopts — необязательные параметры вывода, которые управляют представлением результата (readonly, dadatype и др., см. справку Maple). Удалим с помощью этих команд две строки из преобразованной матрицы C:

```
> DeleteRow(convert(C,Matrix),[1,3]);
```

$$[1 \quad 2 \quad 3]$$

Расширить матрицу M можно при помощи команды extend(M,m,n,x); здесь m — число добавляемых строк, n — число добавляемых столбцов, x — значение заполнителя. Для копирования матрицы A в матрицу B начиная с элемента с номером i,j используется команда copyinto(A,B,i,j). При помощи команды concat(M1,M2) можно получить новую матрицу, являющуюся горизонтальным слиянием двух матриц M1 и M2, имеющих одинаковое число строк. Для вертикального слияния матриц A и B используется команда stack(A,B). Например:

```
> G:=concat(C,F);
```

$$G := \begin{bmatrix} a & b & c & 0 & a & F_{1,3} \\ 1 & 2 & 3 & -a & 0 & F_{2,3} \\ x & y & z & -F_{1,3} & -F_{2,3} & 0 \end{bmatrix}$$

В пакете linalg для извлечения (без удаления) из матрицы M строки или столбца с номером i используются соответственно команды row(M,i) и col(M,i). Для формирования подматрицы, включающей элементы столбцов с номерами от i1 до i2 и строк с номерами от j1 до j2, предусмотрена команда submatrix(M,i1..i2,j1..j2). Команда subvector(v,i..j) выделяет вектор с элементами v[i], ...,v[j] из вектора v. Получить минор матрицы M для элемента с индексами i,j можно командой minor(M,i,j). В пакете LinearAlgebra для работы с переменными типа Matrix и Vector существуют аналоги перечисленных команд: Row, Column, SubMatrix, SubVector, Minor.

Приведем пример выделения подматрицы:

```
> submatrix(G,2..3,3..4);
```

$$\begin{bmatrix} 3 & -a \\ z & -F_{1,3} \end{bmatrix}$$

А теперь выделим минор матрицы:

```
> Minor(A1,3,3);
```

$$\begin{bmatrix} 0 & -a+c \\ a-c & 0 \end{bmatrix}$$

Опишем некоторые команды пакета linalg, позволяющие оперировать со строками и столбцами матрицы. Так, команда addcol(M,i1,i2,expr) формирует новую матрицу, получаемую из матрицы M прибавлением к столбцу с номером i2 столбца с номером i1, умноженного на выражение expr. Аналогичная команда для строк имеет вид: addrow(M,i1,i2,expr). Для умножения столбца или строки матрицы M с номером i на выражение expr применяются соответственно команды mulcol(M,i,expr) и mulrow(M,i,expr). Чтобы переставить местами строки матрицы M с номерами i1 и i2, нужно выполнить команду swaprow(M,r1,r2), а для перестановки столбцов используется команда swapcol(A, c1, c2).

Основные матричные и векторные операции

Перейдем к командам, которые реализуют основные векторные и матричные операции. Начнем с самых простых и часто необходимых операций линейной алгебры — действий с самими матрицами и векторами. Вначале рассмотрим команды пакета linalg, а затем их аналоги из пакета LinearAlgebra.

Для выполнения арифметических операций сложения, вычитания и умножения удобно использовать команду evalm, кроме того, есть команды для отдельных операций. Сложение двух матриц (векторов) A и B одинаковой размерности с помощью этой команды выглядит следующим образом: evalm(A+B). Умножить матрицу A на матрицу (вектор) B можно двумя способами: multiply(A,B) или evalm(A&*B). Проиллюстрируем сказанное примерами:

```
> evalm(C+F);
```

$$\begin{bmatrix} a & b+a & c+F_{1,3} \\ 1-a & 2 & 3+F_{2,3} \\ x-F_{1,3} & y-F_{2,3} & z \end{bmatrix}$$

```
> multiply(C,u);
```

$$[a\,u_1 + b\,u_2 + c\,u_3,\ u_1 + 2\,u_2 + 3\,u_3,\ x\,u_1 + y\,u_2 + z\,u_3]$$

Возведение матрицы M в степень n осуществляется командой evalm(M^n). Обратную матрицу к матрице M можно вычислить также двумя способами: inverse(M) или evalm(1/M). Транспонировать матрицу M можно при помощи команды transpose(M). Вычислить эрмитову транспонированную матрицу можно командой htranspose(A). Для вычисления сопряженной матрицы используется команда adjoint(M), у которой есть короткая форма записи adj(M). Приведем несколько примеров:

```
> adjoint(C);
```

$$\begin{bmatrix} 2\,z - 3\,y & -b\,z + c\,y & 3\,b - 2\,c \\ -z + 3\,x & a\,z - c\,x & -3\,a + c \\ y - 2\,x & -a\,y + b\,x & 2\,a - b \end{bmatrix}$$

```
> transpose(F);
```

$$\begin{bmatrix} 0 & -a & -F_{1,3} \\ a & 0 & -F_{2,3} \\ F_{1,3} & F_{2,3} & 0 \end{bmatrix}$$

Вычислить определитель матрицы M можно командой det(M) пакета linalg или при помощи команды стандартной библиотеки Det(M). Напомним, что для получения результата в последнем случае нужно дополнительно дать команду value. Для получения числа обусловленности используется команда cond(M), а для вычисления следа — команда trace(M). Ранг матрицы M вычисляется командой rank(M). Например:

```
> det(F);
```

$$0$$

```
> rank(G);
```

$$3$$

```
> trace(C);
```

$$a + 2 + z$$

Арифметические операции с матрицами и векторами типа Matrix и Vector, с которыми работает пакет LinearAlgebra, определяются как обычные операции с числами. Для сложения двух матриц A и B в этом случае достаточно выполнить команду A+B, для умножения — A*B и т. д. Если к матрице A прибавляется скалярная величина s, то она добавляется к диагональным элементам. Приведем примеры:

```
> Al:=Matrix(3,[[1,2,3],[4,5,6],[a,b,c]]);
```

$$Al := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ a & b & c \end{bmatrix}$$

```
> Al^2;
```

$$\begin{bmatrix} 9 + 3\,a & 12 + 3\,b & 15 + 3\,c \\ 24 + 6\,a & 33 + 6\,b & 42 + 6\,c \\ a + 4\,b + c\,a & 2\,a + 5\,b + c\,b & 3\,a + 6\,b + c^2 \end{bmatrix}$$

```
> Bl:=Al+13;
```

$$Bl := \begin{bmatrix} 14 & 2 & 3 \\ 4 & 18 & 6 \\ a & b & c+13 \end{bmatrix}$$

```
> Al-Bl;
```

$$\begin{bmatrix} -13 & 0 & 0 \\ 0 & -13 & 0 \\ 0 & 0 & -13 \end{bmatrix}$$

Кроме этих элементарных операций существуют расширенные варианты команд сложения и умножения матриц. Команда `Add(A, B, c1, c2, ip, outopts)` производит сложение матриц `A` и `B` со скалярными множителями `c1` и `c2`: `c1*A+c2*B`. Если параметр `ip` имеет вид `inplace=true`, то результат размещается в матрице `A`. О параметрах `outopts`, которые позволяют определять представление результата, смотри описание команды `Matrix` в этой главе и в справке Maple. Умножить две матрицы `A` и `B` можно при помощи команды `Multiply(A, B, ip, outopt)`. Для поэлементного сложения двух матриц или векторов одинаковых размерностей `A` и `B` существуют соответственно команды `MatrixAdd(A, B, c1, c2, ip, outopts)` и `VectorAdd(A, B, c1, c2, ip, outopts)`. Пример сложения с множителями двух матриц:

```
> Add(Al,Bl,2,-2);
```

$$\begin{bmatrix} -26 & 0 & 0 \\ 0 & -26 & 0 \\ 0 & 0 & -26 \end{bmatrix}$$

Для применения к элементам матриц `A` и `B` функции `f`, в качестве которой может фигурировать знак операции или имя процедуры, существует команда `Zip(f, A, B, ext)`, где `ext` — значение, которое заменяет при обращении недостающие элементы матриц в случае их разной размерности. Продемонстрируем использование этой команды:

```
> A := <<1,2,3>|<4,5,6>>; B := <<1,2>|<3,4>|<5,6>>;
  f:=(x,y)->x^y;
```

$$A := \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

$$B := \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

```
> Zip( "f", A, B);
```

$$\begin{bmatrix} 1 & 64 \\ 4 & 625 \end{bmatrix}$$

```
> Zip("f",A,B,Pi);
```

$$\begin{bmatrix} 1 & 64 & \pi^5 \\ 4 & 625 & \pi^6 \\ 3^\pi & 6^\pi & \pi^\pi \end{bmatrix}$$

В заключение описания основных матричных и векторных операций перечислим некоторые команды пакета LinearAlgebra без подробного описания их параметров:

- Adjoint — вычисление сопряженной матрицы;
- ConditionNumber — число обусловленности матрицы;
- Determinant — определитель матрицы;
- Rank — ранг матрицы;
- Trace — след квадратной матрицы;
- Transpose — транспонированная матрица;
- HermitianTranspose — Эрмитова транспонированная матрица.

Решение задач линейной алгебры

Этот раздел начнем с изложения команд, позволяющих находить спектр квадратной матрицы M. В стандартной библиотеке для поиска собственных чисел и собственных векторов числовой матрицы существует команда с отложенным исполнением

```
Eigenvals (M,vects);
```

здесь M — квадратная числовая матрица, а vects — необязательный параметр, наличие которого говорит о том, что кроме собственных чисел вычисляются и собственные векторы. Результатом действия команды являются собственные числа, а соответствующие собственные векторы будут находиться в колонках матрицы vects. Приведем пример обращения к данной команде:

```
> AA:=array(1..2,1..2,[[23/25,36/25],[36/25,2/25]]):
> vv:=evalf(Eigenvals(AA,W)); eval(W);
```

$$vv := [-1.000000000, 2.000000000]$$

$$\begin{bmatrix} .6000000000 & -.8000000000 \\ -.8000000000 & -.6000000000 \end{bmatrix}$$

Для исследования спектра символьной матрицы можно использовать команды из пакета linalg. Для вычисления собственных чисел матрицы M используется команда eigenvals(M), результатом ее действия является массив, содержащий собственные числа. Для поиска собственных чисел и собственных векторов применяется команда eigenvects(M), причем результат получается в виде массива, каждая строка которого состоит из собственного числа, его кратности и соответствующего собственного вектора. Поясним сказанное примером:

```
> CC:=array(1..3,1..3,[[x,0,y],[x,y,0],[y,0,x]]):
```

$$CC := \begin{bmatrix} x & 0 & y \\ x & y & 0 \\ y & 0 & x \end{bmatrix}$$

```
> linalg[eigenvects](CC);
```

$$[y, 1, \{[0, 1, 0]\}], [y + x, 1, \{[1, 1, 1]\}], \left[-y + x, 1, \left\{\left[\frac{-2\,y + x}{x}, 1, -\frac{-2\,y + x}{x}\right]\right\}\right]$$

В пакете LinearAlgebra также существуют команды для вычисления собственных чисел и векторов. Собственные числа матрицы A можно найти при помощи команды

```
Eigenvalues (A,C,imp,out,outopts)
```

где C — матрица для полной спектральной задачи, параметр imp определяет тип результата вычисления корней характеристического уравнения (implicit — решения представляются через RootOf), а параметр out задает формат результата (output= Vector[row] — вектор-строка, output = Vector[column] — вектор-столбец). Для вычисления собственных векторов матрицы A применяется команда Eigenvectors(A,C,imp, out,outopts). Содержание параметра imp совпадает с описанным для предыдущей команды, а параметр out может принимать значения output= values, vectors или list. О параметрах outopts, которые позволяют изменять представление результата, смотри описание команды Matrix. Применим описанные команды к вычислению собственных значений матрицы CC из предыдущего примера, для чего предварительно преобразуем ее тип:

```
> Eigenvalues(convert(CC,Matrix),output=Vector[row],
             outputoptions=[readonly=true]);
```

$[y, y+x, -y+x]$

```
> Eigenvectors(convert(CC,Matrix),output=[vectors,values]);
```

$$\begin{bmatrix} 0 & \frac{-2\,y+x}{x} & 1 \\ 1 & 1 & 1 \\ 0 & -\frac{-2\,y+x}{x} & 1 \end{bmatrix}, \begin{bmatrix} y \\ -y+x \\ y+x \end{bmatrix}$$

Для вычисления характеристического многочлена матрицы M относительно переменной λ в пакете linalg используется команда charpoly(M,lambda), а в пакете LinearAlgebra — CharacteristicPolynomial(M, lambda). Выяснить положительную или отрицательную определенность матрицы можно при помощи команды definite (M,kind) пакета linalg. Здесь M — квадратная матрица, а kind — параметр, который может принимать значения "positive_def", "positive_semidef", "negative_def" и "negative_semidef". Аналог этой команды в пакете LinearAlgebra — IsDefinite(M, q), где параметр q имеет вид: query=kind. Для проверки ортогональности и унитарности матрицы M в пакете LinearAlgebra предусмотрены соответственно команды IsOrthogonal(M) и IsUnitary(M), а подобие двух матриц M и N выясняется командой IsSimilar(M,N). Проверить ортогональность матрицы M можно и командой linalg[orthog(M)]. Результатом действия последних команд будет булевская константа (true или false). Пример:

```
> A := LinearAlgebra[DiagonalMatrix]([-5,0,-1]);
```

$$A := \begin{bmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

```
> LinearAlgebra[IsDefinite](A,'query'='negative_semidef');
```

true

Команда MinimalPolynomial(M,lambda) пакета **LinearAlgebra** вычисляет минимальный полином матрицы по переменной λ. Ядро матрицы M вычисляется при помощи команд linalg[kernel](M) или LinearAlgebra[NullSpace](M). Например:

```
> F:=Matrix(3,shape=antisymmetric):
> F[2,3]:=a: F[1,2]:=b: F[1,3]:=c: F;
```

$$\begin{bmatrix} 0 & b & c \\ -b & 0 & a \\ -c & -a & 0 \end{bmatrix}$$

В следующем примере используем обращение к команде пакета LinearAlgebra как к элементу модуля:

```
> LinearAlgebra:-NullSpace(F);
```

$$\left\{\begin{bmatrix} -\frac{a}{c} \\ 1 \\ -\frac{b}{c} \end{bmatrix}\right\}$$

В Maple реализованы практически все алгоритмы приведения матриц к различным специальным формам. Так, для приведения матрицы M к жордановой форме используются команды linalg[jordan](M) и LinearAlgebra[JordanForm](M). Результатом вызова команды gausselim(M) будет матрица, приведенная к треугольному виду. Применить к матрице M алгоритм гауссова исключения без деления можно командой ffgausselim(M). Команда приведения матрицы M к треугольному виду при помощи алгоритма Гаусса–Жордана называется gaussjord. Последние три команды входят в пакет **linalg**. Чтобы получить Эрмитову форму матрицы M, элементы которой зависят от переменной x, нужно обратиться к команде linalg[hermite](M,x) или LinearAlgebra[HermiteForm](M,x). Приведем примеры:

```
> C:=array(1..3,1..3,[[a,b,c],[1,2,3],[x,y,z]]):
> LinearAlgebra[HermiteForm](convert(C,Matrix),x);
```

$$\begin{bmatrix} 1 & 0 & -\frac{3b-2c}{2a-b} \\ 0 & 1 & \frac{3a-c}{2a-b} \\ 0 & 0 & \frac{-bz+cy+2az-2cx-3ay+3bx}{3b-2c} \end{bmatrix}$$

```
> linalg[gaussjord](C);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> linalg[ffgausselim](C);
```

$$\begin{bmatrix} a & b & c \\ 0 & 2a-b & 3a-c \\ 0 & 0 & 2za-zb-2xc-3ya+yc+3xb \end{bmatrix}$$

Отметим, что для работы с символьными матрицами команда ffgausselim предпочтительнее других, поскольку не производит нормировку элементов и исключает возможные ошибки, связанные с делением на нуль.

Кроме того, в пакете **LinearAlgebra** есть команды LUDecomposition и QRDecomposition, которые реализуют LU и QR преобразования соответственно. Результатом работы этих команд является набор матриц. Пример:

```
> FF:=Matrix([[1,2,3],[1,a,7],[2,6,9]]):
> LUDecomposition(FF);
```

$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\1 & 1 & 0\\2 & 2\frac{1}{a-2} & 1\end{bmatrix}, \begin{bmatrix}1 & 2 & 3\\0 & a-2 & 4\\0 & 0 & \frac{3a-14}{a-2}\end{bmatrix}$$

Следующая группа команд позволяет изучать векторное пространство, порождаемое матрицей M. Так, для определения размерности векторного пространства, порожденного столбцами матрицы в пакете **linalg**, используется команда colspace(M), а соответствующая команда для строк называется rowspace(M). Результатом работы этих команд являются векторы, задающие базис пространства. Аналогичные команды пакета **LinearAlgebra** называются ColumnSpace(M) и RowSpace(M). Например:

```
> fun:=(i,j)->x^i/y^j: B:=matrix(2,3,fun);
```

$$B := \begin{bmatrix}\frac{x}{y} & \frac{x}{y^2} & \frac{x}{y^3}\\ \frac{x^2}{y} & \frac{x^2}{y^2} & \frac{x^2}{y^3}\end{bmatrix}$$

```
> linalg[colspace](B);
```

$$\{[1, x]\}$$

```
> LinearAlgebra[RowSpace](convert(B,Matrix));
```

$$\left[\left[1, \frac{1}{y}, \frac{1}{y^2}\right]\right]$$

Для решения систем линейных уравнений в пакете **linalg** имеются команды, отличные от команды solve из стандартной библиотеки. Решение системы линейных алгебраических уравнений Mx=B, где M — матрица, а B — вектор или матрица правых частей, находится командами linsolve(M,B) или leastsqrs(M,B) (метод наименьших квадратов). Аналогичные команды пакета **LinearAlgebra** называются соответственно LinearSolve и LeastSquares. Приведем пример решения системы линейных уравнений:

```
> CC:=array(1..3,1..3,[[x,0,y],[x,y,0],[y,0,x]]):
> v:=array(1..3,[1,1,1]):
> linalg[linsolve](CC, v);
```

$$\left[\frac{1}{y+x}, \frac{1}{y+x}, \frac{1}{y+x}\right]$$

В заключение приведем некоторые команды пакета linalg для работы с полиномами и уравнениями:

- sylvester(pol1,pol2,x) — вычисление матрицы Сильвестра для полиномов pol1 и pol2, зависящих от переменной x;
- bezout(pol1,pol2,x) — формирование матрицы Безу двух полиномов;
- genmatrix(eqns,vars) — формирование матрицы системы для набора уравнений eqns по переменным vars.

Векторный анализ

Теперь перейдем к векторным операциям. Нормировать вектор v можно при помощи команды normalize(v) пакета linalg или команды Normalize(V,p) пакета LinearAlgebra, где параметр p задает вид нормы. Например:

```
> Normalize(<1|0|3>,Euclidean);
```

$$\left[\frac{1}{10}\sqrt{10}, 0, \frac{3}{10}\sqrt{10}\right]$$

```
> Normalize(<1|0|3>,inplace);
```

$$\left[\frac{1}{3}, 0, 1\right]$$

Чтобы вычислить угол между двумя векторами v и u, нужно выполнить команду linalg[angle](v,u) или LinearAlgebra[VectorAngle](u,v). При помощи команды GramSchmidt ([v1,v2,...]) можно сформировать ортогональный базис векторного пространства, генерируемого линейно-независимыми числовыми векторами v1, v2, ... Команда basis([v1,v2,...vk]) пакета linalg определяет базис для набора векторов v1, v2, ..., vk. В пакете LinearAlgebra существуют три команды для определения базиса векторного пространства: Basis(([v1,v2,...vk]) — базис векторов, SumBasis(VS) — базис прямой суммы наборов векторов VS, IntersectionBasis(VS) — базис пересечения наборов векторов VS.

Приведем иллюстрирующий некоторые из этих команд пример:

```
> with(LinearAlgebra): v1:=<1|2|3>: v2:=<-1|2|3>:
> v3:=<11|12|13>: v4:=<1|1|1>: bas:=Basis([v1,v2,v3,v4]);
```

$$bas := [[1, 2, 3], [-1, 2, 3], [11, 12, 13]]$$

```
> GramSchmidt(bas);
```

$$\left[[1, 2, 3], \left[\frac{-13}{7}, \frac{2}{7}, \frac{3}{7}\right], \left[0, \frac{30}{13}, \frac{-20}{13}\right]\right]$$

```
> IntersectionBasis([[v1,v2],[v3,v4]]);
```

$$[[1, 2, 3]]$$

Скалярное произведение двух векторов u и v вычисляется командами linalg[dotprod](u,v) и LinearAlgebra[DotProduct](u,v), а векторное произведение — командами linalg[crossprod](u,v) и LinearAlgebra[CrossProduct](u,v). Для вычисления нормы векторов и матриц используются команды linalg[norm](M, normname) и LinearAlgebra [Norm](M,normtype). В качестве параметра normname может фигурировать 1, 2, "infinity" или "frobenius", а параметра normtype — положительное целое, infinity, Euclidean или Frobenius. Приведем примеры:

```
> w:=linalg[crossprod]([a,b,c],[d,e,f]);
```

$$w := [bf - ce, cd - af, ae - bd]$$

```
> simplify(linalg[dotprod](w,[a,b,c])); ww:=%:
```

$\overline{(a)}\,b\,f-\overline{(a)}\,c\,e+\overline{(b)}\,c\,d-\overline{(b)}\,a\,f+\overline{(c)}\,a\,e-\overline{(c)}\,b\,d$

Обратим внимание на то, что в результате упрощения смешанного произведения с двумя одинаковыми векторами не получилось нуля. Это объясняется тем, что Maple по умолчанию считает все математические переменные комплексными, а в этом случае искомое выражение не равно нулю. Теперь определим все переменные в выражении действительными при помощи команды assume и снова упростим выражение:

```
> assume(a,real); assume(b,real); assume(c,real);
> simplify(ww);
```

0

Перейдем к дифференциальным операторам векторного анализа пакета linalg. Чтобы вычислить градиент скалярной функции f, зависящей от переменных X, нужно выполнить команду grad(f,X). Дивергенцию векторной функции F, зависящей от переменных X, можно получить при помощи команды diverge(F,X). Например:

```
> f:=x^2+y^2+sin(z): grd:=grad(f,[x,y,z]);
```

$grd := [2\,x, 2\,y, \cos(z)]$

```
> diverge(grd,[x,y,z]);
```

$4 - \sin(z)$

И еще три команды для важных дифференциальных операторов:

- curl(v,x) — вычисление ротора трехмерного вектора v по трем переменным x;
- laplacian(f,x) — вычисление лапласиана функции f по переменным x;
- jacobian(v,x) — вычисление матрицы Якоби для вектора v по переменным x.

Дадим примеры использования этих команд:

```
> curl(grd,[x,y,z]);
```

$[0, 0, 0]$

```
> laplacian(f,[x,y,z]);
```

$4 - \sin(z)$

```
> jacobian(grd,[x,y,z]);
```

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -\sin(z) \end{bmatrix}$$

Преобразования в операторной форме

В Maple можно определять абстрактные, не заданные явно математические операторы и работать с ними. Для определения оператора и его свойств используется команда define:

```
define (oper, property1, property2, ...)
```

Здесь oper — имя оператора, property1, property2, ... — его свойства, в качестве которых могут фигурировать уравнения, определяющие правила действия операто-

ра или зарезервированные служебные слова (associative, commutative, linear, multilinear, orderless, flat, identity, zero, diff). Добавление свойств для уже определенного оператора осуществляется командой

```
definemore (oper, property1, property2, ...)
```

Теперь рассмотрим параметры и способы задания абстрактных операторов. С помощью уравнений определяются действия, производимые оператором, или специальные значения, которые оператор может принимать для входных данных специального вида. При описании действий можно определять также тип переменных оператора. Пример:

```
> define(p,p(h(i::integer*x::algebraic),
  h(j::integer*x::algebraic))=h((i+j)*x)+h((i-j)*x));
> p(h(2*x),h(378*x));
```

$h(380\,x) + h(-376\,x)$

Описатель linear означает то, что оператор линеен по первой переменной, а в случае описателя multilinear оператор принимается линейным по всем переменным. Приведем пример определения линейных операторов:

```
> define(g,linear,g(1)=S);
> g(3*t-2);
```

$3\,g(t) - 2\,S$

```
> define(f,multilinear,f(-b::algebraic,a::algebraic)=-f(b,a));
> f(-a+c,b);
```

$-f(a, b) + f(c, b)$

```
> f(a-c,-b+a);
```

$-f(a, b) + f(a, a) + f(c, b) - f(c, a)$

```
> f(g(a-2),-g(a+b));
```

$-f(g(a), g(a)) - f(g(a), g(b)) + 2\,f(S, g(a)) + 2\,f(S, g(b))$

Описатель associative или его синоним flat означает, что оператор является ассоциативным, то есть f(x,f(y,z)) = f(f(x,y),z) = f(x,y,z). Пример:

```
> define(a,associative); a(x,a(y),a(a(a(z))));
```

$a(x, y, z)$

Если в описании оператора присутствует orderless, то порядок аргументов во внимание не принимается. Пример:

```
> define(h,orderless);
> h(a,b)-h(b,a);
```

0

С помощью описателя zero можно определить нуль оператора, а обратный к описываемому оператор задается с помощью параметра inverse. Пример:

```
> define("&v",inverse=g,zero=y);
> &v g(y); &v %;
```

y

0

Описатель diff предназначен для определения производной оператора. Например:

```
> define(p,diff(p(x),x)=q);
  diff(p(i*x),x);
```

$q\ i$

```
> int(p(x),x);
```

$$x\ p(x) - \frac{1}{2} x^2 q$$

Отметим, что при выполнении команд свойства операторов проверяются в порядке их определения командами define и definemore. Поясним это двумя примерами, которые отличаются только последовательностью определения свойств оператора h. В первом случае Maple выведет сообщение об ошибке, а во втором случае свойства определены правильно:

```
> define(h,h(a::algebraic,b::algebraic)=0);
> definemore(h,h(a::algebraic,a::algebraic)=1);
Error, (in canonic) left hand side of equational property "0 = 1" does not contain function name h

> restart;define(h,h(a::algebraic,a::algebraic)=1);
> definemore(h,h(a::algebraic,b::algebraic)=0);
> h(x,y); h(x^2,x^2);
```

0

1

В заключение напомним, что кроме команды define для алгебраических выкладок в абстрактном (операторном) виде существуют пакеты group и Ore_algebra, однако мы здесь их описывать не будем.

6 ГЛАВА Графика Maple

Одним из интересных и эффектных применений описываемых в книге пакетов является использование их графических возможностей при решении различных задач: для визуализации результатов исследований, графической интерпретации данных и т. д. Если даже отбросить все другие возможности Maple и изучить только графические команды, то эти знания позволят пользователю эффективно применять возможности компьютерной графики — от рисования простого графика функции до создания мультфильмов. Перечислим основные графические возможности пакета Maple:

- рисование кривых, определяемых функциями одной действительной переменной, параметрически заданных и неявно определенных кривых;
- рисование векторных и градиентных полей функции двух переменных, линий уровня функции;
- изображение данных, заданных координатами точек, как на плоскости, так и в проекциях трехмерного пространства;
- построение поверхности, определяемой функцией двух действительных переменных, параметрически заданных кривых и поверхностей в трехмерном пространстве, рисование неявно заданных поверхностей;
- конструирование двумерных и трехмерных графических объектов (окружностей, сфер, отрезков, прямоугольников и т. д.) и манипуляции с ними;
- двумерная и трехмерная мультипликация с использованием графических объектов.

Исходными данными для построения графических образов могут быть конструкции Maple (функции, массивы, графические структуры), а также результаты вычислений, полученные при помощи других программ и записанные в текстовых файлах. В качестве параметров при вызове графических команд указываются выводимые объекты (функция, набор точек, множество), интервалы изменения переменных и параметры вывода, управляющие видом изображения.

В Maple имеется богатый набор команд двумерной и трехмерной графики. Эти команды находятся в основном в пакетах plots и plottools. Первый из них содержит

команды построения различных графических объектов, а второй предназначен для манипуляций с такими объектами. Ряд графических команд, ориентированных на решение специальных задач, находится в других библиотеках, и о некоторых из них сказано в соответствующих главах (например, пакет DEtools для исследования дифференциальных уравнений или статистический пакет stats).

Наиболее универсальные и часто используемые команды рисования двумерных графиков plot и поверхностей plot3d графической библиотеки plots доступны пользователю по умолчанию. Перед обращением к другим командам нужно подключить библиотеку командой

```
> with(plots);
Warning, the name changecoords has been redefined
[animate, animate3d, animatecurve, changecoords, complexplot, complexplot3d, conformal,
contourplot, contourplot3d, coordplot, coordplot3d, cylinderplot, densityplot, display,
display3d, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot, implicitplot3d, inequal,
listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot,
matrixplot, odeplot, pareto, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d,
polyhedra_supported, polyhedraplot, replot, rootlocus, semilogplot, setoptions, setoptions3d,
spacecurve, sparsematrixplot, sphereplot, surfdata, textplot, textplot3d, tubeplot]
```

В результате становятся доступными команды из этого списка. Например, важные команды display и display3d позволяют не только перерисовывать графические объекты, но и выводить на одном рисунке несколько разнотипных объектов.

Другим способом вызова конкретной команды (без подключения всего пакета) является следующая конструкция:

```
plots[имя команды](параметры).
```

Для подключения пакета команд манипуляции с графическими объектами надо выполнить следующую команду:

```
> with(plottools);
[arc, arrow, circle, cone, cuboid, curve, cutin, cutout, cylinder, disk, dodecahedron,
ellipse, ellipticArc, hemisphere, hexahedron, homothety, hyperbola, icosahedron, line,
octahedron, pieslice, point, polygon, project, rectangle, reflect, rotate, scale, semitorus,
sphere, stellate, tetrahedron, torus, transform, translate, vrml]
```

Результатом выполнения графической команды будет построение рисунка. В среде Windows рисунок помещается в текст Maple-документа. При выделении построенного рисунка возникает специальное графическое меню. С помощью пунктов этого меню можно интерактивно изменять ракурс, цвет и другие параметры построенного изображения. Однако из меню доступно только частичное управление видом изображения, а полное управление реализуется при помощи управляющих параметров графической команды. Подробнее о пунктах меню можно прочитать в Приложении. Параметры могут присутствовать в любых графических командах, которые схематически можно представить в следующем виде:

```
Имя_Графической_Команды(Обязательные параметры, Управляющие параметры)
```

Результат работы графической команды можно присвоить переменной, при этом вывода рисунка не происходит. Чтобы рисунок появился, достаточно дать команду просмотра переменной, содержащей рисунок. Это позволяет подготовить несколько графических объектов и затем составить из них единый рисунок с помощью команд display и display3d.

Обратим внимание на то, что при обращении к графическим командам в выражениях, которые задают рисуемые объекты, не должно быть неопределенных переменных. Если это не так, то Maple выведет сообщение о невозможности построения графика:

```
Plotting error, empty plot
```

Работа графических команд происходит в три этапа. Вначале анализируется правильность введенной команды. Затем параметры трансформируются в графические структуры. Эти структуры являются аппаратно-независимыми и определяют объект, который должен быть изображен. Завершает работу вывод графической структуры на изображающее устройство. По умолчанию в Windows-версии таким устройством является экран монитора, но при помощи команд interface или plotsetup (см. главу 7 «Программирование в Maple») его можно переопределить, указав другие устройства (например, файл).

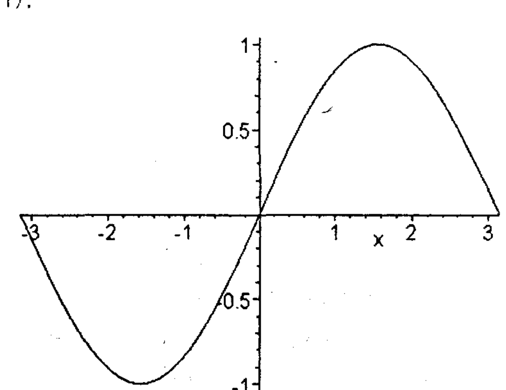
Ниже рассмотрены основные команды, параметры и структуры двумерной и трехмерной графики. Для лучшего понимания работы графических команд нужно ознакомиться с главами, посвященными графическим структурам. Эта информация будет особенно полезна тем, кто собирается создавать собственные программы с использованием графического вывода.

Перед тем как перейти к подробному изложению, приведем без комментариев самый простой вариант рисования графика функции одной переменной f(x) на интервале a<x<b:

```
plot(f(x),x=a..b);
```

Пример:

```
> plot(sin(x),x=-Pi..Pi);
```



Двумерная графика

Представим описание возможностей двумерной графики, начав с рассмотрения ее структур. Затем разберем пакет plottools, команды которого позволяют работать с графическими объектами. После этого перечислим основные параметры команд двумерной графики. Потом подробно разберем команду plot. И закончим описа-

ние перечнем специальных команд, предназначенных для получения большинства стандартных графических математических построений.

Структуры двумерной графики

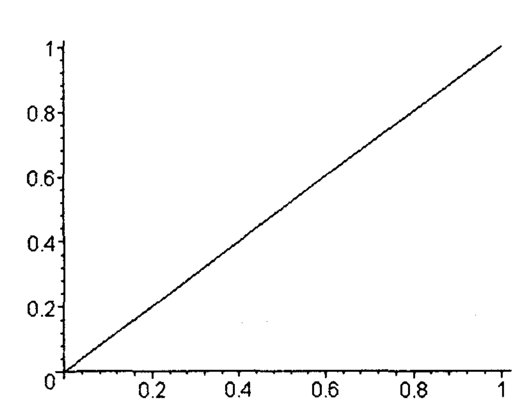
Графические команды преобразуют входные данные в графические структуры и выводят их на экран при помощи команды PLOT. Эти структуры состоят из небольшого числа элементарных объектов, о которых мы скажем ниже, и являются составной частью стандартной библиотеки. Начнем с примера графической структуры, которая создается уже знакомой командой plot. Мы присвоим переменной pic результат выполнения команды plot, которая соединяет две точки с координатами (0,0) и (1,1):

```
> pic:=plot([[0,0],[1,1]]);
```

pic := PLOT(CURVES([[0., 0.], [1., 1.]], COLOUR(*RGB*, 1.0, 0., 0.)),
AXESLABELS("", ""), VIEW(*DEFAULT*, *DEFAULT*))

В результате переменной pic присвоена графическая структура, содержащая информацию об отрезке (CURVES), цвете (COLOUR) и др. С помощью графических структур можно конструировать собственные команды. Для изображения графической структуры достаточно к ней обратиться. Например:

```
> pic;
```



В состав структуры входят описания элементарных объектов и параметры.

Перечислим двумерные графические объекты:

- POINTS([x1,y1],[x2,y2],...) — набор точек с координатами [x1,y1], [x2,y2], ...;
- CURVES([[x11,y11],...,[x1n,y1n]],...,[[xm1,ym1],...,[xmn,ymn]]) — m наборов по n точек, определяющих незамкнутые кривые. Точки каждого из наборов будут последовательно соединены отрезками;
- POLYGONS([[x11,y11],...,[x1n,y1n]],...,[[xm1,ym1],...,[xmn,ymn]]) — m наборов по n точек, определяющих замкнутые кривые. Точки каждого из наборов будут последовательно соединены отрезками, причем последняя точка набора соединяется с первой;
- TEXT([x,y],"str",horizontal,vertical,font) — графический вывод текста str начиная с позиции, заданной координатами [x,y]. Необязательный параметр horizontal

определяет вид горизонтального выравнивания и может принимать значение ALIGNLEFT или ALIGNRIGHT. Способ вертикального выравнивания задает необязательный параметр vertical (ALIGNABOVE или ALIGNBELOW). Еще один необязательный параметр font управляет видом и размером шрифта (см. примеры и справку Maple).

Для вывода графических объектов str1,...,strn на экран используется команда стандартной библиотеки

```
PLOT({str1,...,strn});
```

Результатом действия этой команды будет графическая структура.

Приведем пример. Сформируем два графических объекта: один — для незамкнутой кривой, другой — для набора точек, а затем нарисуем их:

```
> str1:=CURVES([[0,0],[1,0.1],[1,0.4],[0.1,0.5]]):
> str2:=POINTS([0.5,0.2],[0.4,0.4],[0.7,0.3]):
> PLOT(str1,str2,AXESSTYLE(NONE));
```



В последней команде использована конструкция AXESSTYLE(NONE). Это один из параметров графических структур, который определяет тип осей координат (в примере — отменяет их). При задании графических структур можно использовать следующие параметры: AXESSTYLE (тип осей координат), STYLE (стиль рисования, точки или линии), LINESTYLE (тип линии), THICKNESS (толщина линии), SYMBOL (тип символа), FONT (тип и размер шрифта), AXESTICKS (разметка осей координат), AXESLABELS (метки на осях координат), VIEW (координаты рисуемой области), SCALING (тип масштабирования), TITLE (заголовок объекта), COLOR (задание цвета) и др. Некоторые примеры приведены ниже; при необходимости более подробной информации пользователь может обратиться к справке пакета по графическим структурам ?plot,structure.

Приведем пример использования графических структур для построения гистограммы, определяемой пятью значениями функции y=x(1-x) на отрезке [0,1]. В этом примере мы постараемся использовать побольше графических объектов и параметров.

В первую очередь определим саму функцию и процедуру-функцию, которая задает четыре вершины трапеции гистограммы:

```
> f:=x->x*(1-x):
> p:=i->[[(i-1)/5,0],[(i-1)/5,f((i-1)/5)],
    [i/5,f(i/5)],[i/5,0] ]:
```

Сформируем графическую структуру set1 из пяти трапеций, окрашенных в различные оттенки серого цвета:

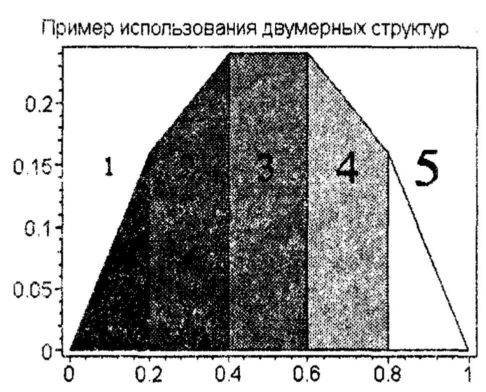
```
> set1:=seq(POLYGONS(p(i),COLOR(RGB,i/5,i/5,i/5)),i=1..5):
```

Графическая структура set2 содержит пять строк текста, каждая из которых задает номера столбца гистограммы, причем размер шрифта последовательно увеличивается:

```
> set2:=seq(TEXT([(i-0.5)/5,0.15],convert(i,string),
  FONT(SYMBOL,10+3*i)),i=1..5):
```

Теперь обе графические структуры выведем на экран, снабдив рисунок заголовком и определив тип осей координат:

```
> PLOT(set1,set2,AXESSTYLE(BOX),
  TITLE("Пример использования двумерных структур")):
```



Двумерные команды пакета plottools

Часто возникает необходимость выполнять различные операции с графическими структурами (вращать, уменьшать и пр.), в частности с результатами графических команд. Такую возможность предоставляют команды из пакета plottools. Кроме того, в этот пакет включены команды, генерирующие простейшие и часто требующиеся графические объекты (окружность, стрелки, кривые и др.). Далее мы перечислим многие команды пакета и проиллюстрируем их действие примерами. Полный список команд можно найти в справке пакета.

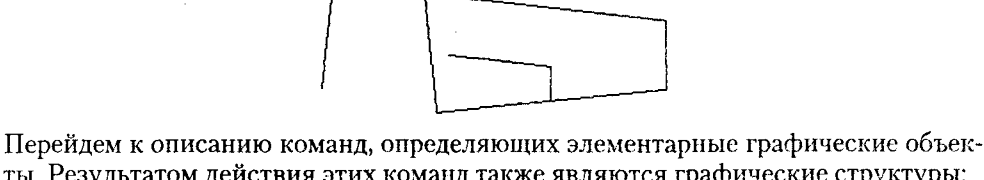
Команды пакета plottools работают только с графическими структурами, а результатом их действия является графический объект. Для изображения графического объекта требуется использовать команду PLOT из ядра Maple или команду display пакета plots. Начнем с команд манипуляций с графическими объектами:

- reflect(GS,[x,y]) — отражение графической структуры GS относительно точки с координатами (x,y). Если за точкой [x,y] в параметрах команды следует точка [x1,y1], то структура отражается относительно прямой, проходящей через эти точки;
- rotate(GS,ang,[x,y]) — повернуть графическую структуру GS относительно точки с координатами (x,y) на угол ang (в радианах);

- project(GS,[pt1,pt2]) — спроецировать графическую структуру GS на прямую, определяемую двумя точками [pt1,pt2]. Здесь pt1,pt2 – векторы из двух координат;
- scale(GS,A,B,[x,y]) — изменить размеры структуры GS в A раз по координате x и в B раз по координате y. В качестве центра масштабирования может быть указана точка с координатами (x,y);
- translate(GS,A,B) — увеличить все координаты в графической структуре GS на величину A по оси x и величину B по оси y.

Теперь приведем пример, в котором вместе с кривой str1 из предыдущего раздела мы выведем уменьшенный в два раза ее вариант, копию кривой, повернутую на 90 градусов, и копию, перемещенную вверх на 2/3:

```
> PLOT(str1,rotate(str1, Pi/2, [0,0]),
 scale(str1, 1/2, 1/2, [0,0]),
 translate(str1, 0, 2/3),
 AXESSTYLE(NONE),SCALING(CONSTRAINED));
```



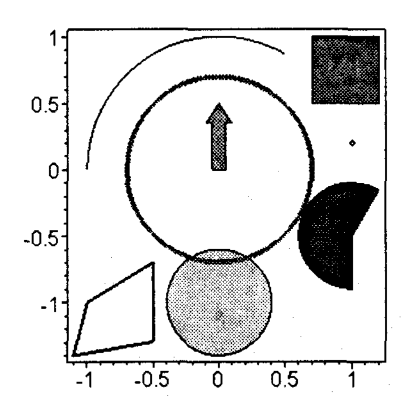
Перейдем к описанию команд, определяющих элементарные графические объекты. Результатом действия этих команд также являются графические структуры:

- arc([x,y], r, a..b) — генерирует дугу окружности с центром в точке [x,y] и радиусом r. Величина угла задается в радианах интервалом a..b;
- arrow([x,y], [v1,v2], wb, wh, hh) — задает графический объект в виде стрелки, [x,y] определяют координаты основания стрелки, вектор [v1,v2] указывает ее направление. Параметры wb и wh задают ширину нижней и верхней частей стрелки соответственно, а hh дает отношение ширины к длине верхней части стрелки;
- circle([x,y], r) — генерирует окружность с центром в точке [x,y] и радиусом r;
- disk([x,y], r) — задает круг с центром в точке [x,y] и радиусом r;
- ellipse([x,y], a, b) — формирует эллипс с центром в точке [x,y] и полуосями a и b;
- line([x1,y1], [x2,y2]) — соединяет линией точки с координатами [x1,y1] и [x2,y2];
- pieslice([x,y], r, a..b) — задает сектор круга с центром в точке [x,y] и радиусом r, угол задается интервалом a..b;
- point(pset) — определяет набор точек. Здесь в качестве параметра pset выступает список, состоящий из точек. Каждая точка дается двумя координатами;
- polygon([[x1,y1],..., [xn, yn]]) — генерирует многоугольник, соединяющий последовательно точки с координатами [x1,y1], ..., [xn, yn]. Последняя точка соединяется с первой. Внутренняя часть многоугольника может быть закрашена цветом, который указывается при помощи параметра color (см. следующий раздел);

- rectangle([x1,y1], [x2,y2]) — определяет прямоугольник с левым верхним углом [x1,y1] и правым нижним углом [x2,y2]. С помощью параметров внутренность прямоугольника может быть закрашена.

Проиллюстрируем действие описанных команд. Обратим внимание, что для задания некоторых объектов использованы параметры, которые будут описаны в следующем разделе. Для вывода графических структур применена команда display:

```
> a := arc([0,0], 1, Pi/3..Pi):
> b:=circle([0,0],0.7,color=red,thickness=4):
> c:= arrow([0,0], [0,0.5], .1, .2, .3, color=green):
> d := disk([0,-1], 0.4, color=yellow):
> e := pieslice([1,-0.5], 0.4, Pi/3..3*Pi/2, color=blue):
> f := point([1,0.2], color=black):
> g:=rectangle([0.7,1.0],[1.2,0.5],color=gold):
> h:=polygon([[-1.1,-1.4],[-1,-1],
  [-0.5,-0.7],[-0.5,-1.3]],thickness=2):
> plots[display](a,b,c,d,e,f,g,h,axes=boxed,
  scaling=constrained);
```



Управляющие параметры двумерной графики

Большинство параметров двумерной графики, за исключением специализированных, применимы для всех графических команд. Они позволяют управлять изображением: детальностью графика, типом выводимых линий и заполнителей, размещением надписей и т. д. Управляющие параметры в командах следуют сразу за обязательными параметрами, а при их отсутствии используются установки по умолчанию (см. справку Maple). Пользователь может переопределить установки на сеанс при помощи команды setoptions(options). Перечислим и прокомментируем основные параметры.

Управление цветом и шрифтами

Для управления цветом и шрифтами применяются следующие команды:

- color=colorvalue — цвет вывода. В качестве colorvalue может выступать одно из следующих зарезервированных в Maple названий цветов: aquamarine, black, blue,

navy, coral, cyan, brown, gold, green, gray, grey, khaki, magenta, maroon, orange, pink, plum, red, sienna, tan, turquoise, violet, wheat, white, yellow. Кроме того, существует возможность определения собственных цветов. Для этого можно использовать параметр графических структур COLOR. Одним из вариантов обращения к ней является следующий: COLOR(RGB,r,g,b). Здесь RGB указывает на то, что цвет задается долями красного, зеленого и синего цветов, а параметры r, g, b указывают доли этих цветов и могут принимать значения от нуля до единицы;

- font=[vfon,vstyle,vsize] — шрифт для вывода текста. Здесь переменная vfon задает имя шрифта (TIMES, COURIER, HELVETICA или SYMBOL), переменная vstyle определяет стиль шрифта (см. справку Maple), а число vsize — размер символа;
- labelfont=l — определяет шрифт, которым выводится текст на осях координат. Обращение аналогично рассмотренному для параметра font.

Параметры, определяющие вид осей координат и заголовка

Определить вид осей координат и заголовка можно при помощи следующих параметров:

- title="Name" — заголовок рисунка;
- titlefont=[vfon,vstyle,vsize] — определяет шрифт, которым выводится текст заголовка. Обращение аналогично рассмотренному для параметра font;
- axes=val — тип выводимых осей координат. Величина val может принимать одно из следующих значений: NORMAL — обычные оси координат, BOXED — график заключается в рамку с нанесенной шкалой, FRAME — оси с центром в левом нижнем углу рисунка, NONE — вывод без нанесения осей;
- xtickmarks=nx — число насечек по оси абсцисс (X);
- ytickmarks=ny — число насечек по оси ординат (Y);
- labels=[str_X, str_Y] — надписи по осям координат. По умолчанию принимаются имена выводимых переменных;
- labeldirections=[labx,laby] — указывает тип размещения меток осей координат (вертикальное или горизонтальное). Величины labx, laby могут принимать значение HORIZONTAL или VERTICAL. По умолчанию принято первое значение;
- legend=stri — комментарий (легенда) к кривой в виде строки stri. Если на графике несколько кривых, то в качестве stri должен фигурировать список из строк.

Параметры, задающие стиль графика и вид линий

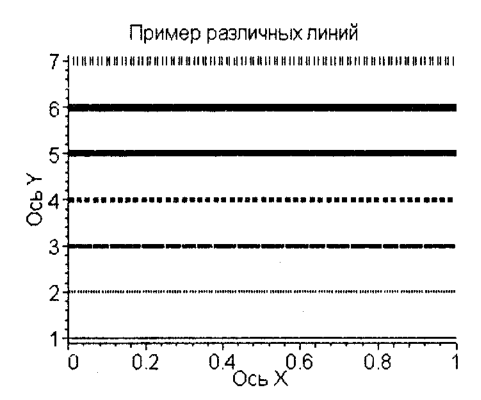
Стиль линий и вид графика можно при помощи следующих параметров:

- style=LINE/POINT — вывод графика линиями или точками. В некоторых двумерных графических командах этот параметр может принимать значения PATCH или PATCHNOGRID;
- thickness=n — толщина линии; n может принимать значения 1,2,3,...;
- linestyle=n — тип выводимой линии (непрерывная, пунктир,...), по умолчанию — непрерывная линия (n=1);

- symbol=s — тип символа (маркера), которым помечаются точки, s может принимать одно из следующих значений: BOX, CROSS, CIRCLE, POINT, DIAMOND;
- symbolsize=n — размер маркера.

Для иллюстрации действия различных параметров мы будем использовать команду рисования графика функции одной действительной переменной plot. Подробно эта важная команда будет рассмотрена в следующем разделе. Приведем пример, иллюстрирующий использование некоторых перечисленных параметров. Нарисуем семь линий различной толщины и различным стилем:

```
> for i from 1 to 7 do p||i:=plot(i,0..1,
  thickness=i,linestyle=i, color=black): od:
> plots[display](seq(p||i,i=1..7),
   title="Пример различных линий",labels=["Ось X","Ось Y"],
   labeldirections=[HORIZONTAL,VERTICAL]);
```



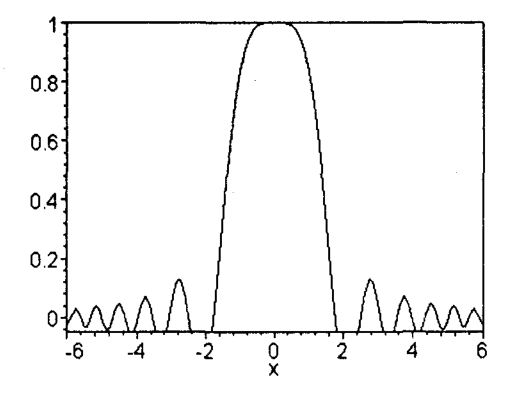
Параметры, определяющие масштабирование, тип координат и разрешение

Масштабирование, тип координат и разрешение можно определить с помощью следующих параметров:

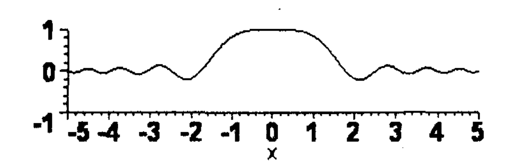
- coords=type — тип координат (polar,cartesian);
- scaling=val — тип масштабирования. Если val принимает значение CONSTRAINED, то график выводится с одинаковым масштабом по осям координат, а при значении UNCONSTRAINED — график масштабируется по размеру графического окна;
- view=[xmin..xmax, ymin..ymax] — область вывода, диапазон изменения горизонтальной переменной задают величины xmin и xmax, а вертикальной — ymin и ymax;
- numpoints=n — число вычисляемых точек графика (по умолчанию 49);
- resolution=n — горизонтальное разрешение дисплея в пикселах (по умолчанию 200).

Приведем два примера, на которых график одной функции выводится с различными параметрами:

```
> plot (sin(x^2)/x^2,x=-6..6,-0.05..1,
  axes=BOXED, scaling=unconstrained);
```




```
> plot(sin(x^2)/x^2,x=-6..6,view=[-5..5,-1..1],axes=FRAMED,
  scaling=constrained,xtickmarks=10,
  axesfont=[HELVETICA,BOLD,12]);
```



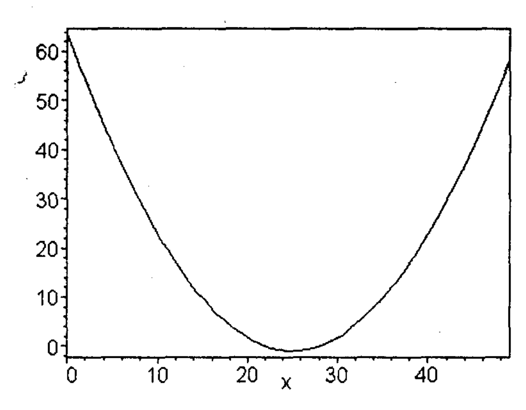
Выше перечислены важнейшие параметры, общие для всех команд двумерной графики. Укажем еще несколько специфических, но важных параметров. Для команд `fieldplot` и `gradplot` существует параметр, задающий тип стрелки для изображения вектора: `arrows= thin` (или `line`, `slim`, `thick`).

Для команд `fieldplot`, `gradplot` и `implicitplot` существует параметр, задающий размеры сетки для вычисления векторного поля и неявно заданной функции на плоскости: `grid=[int1,int2]`, где `int1` — число узлов по оси `x`, `int2` — по оси `y`. По умолчанию принято `grid=[25..25]`.

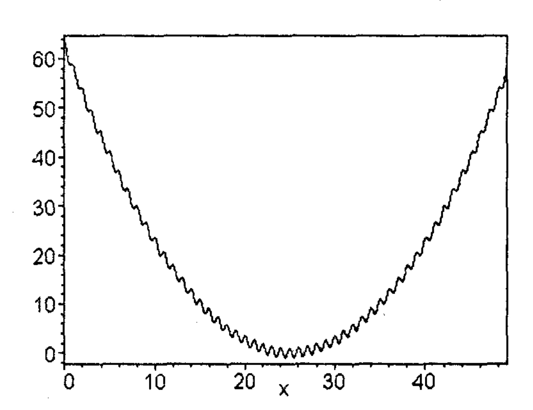
В Windows-версии многие параметры вывода можно переопределить интерактивно через графическое меню (см. конец данной главы). Отметим, что в меню изменяются только внешние атрибуты рисунка (тип осей, вид масштаба и пр.), а некоторые важные параметры можно определить только при задании команды. В графическом меню нельзя вставить или отредактировать надписи, изменить число узлов, поменять цвет линий и т. д.

Неправильное или невнимательное назначение параметров может привести к неточностям. Надо всегда помнить, что построение графика происходит по точкам и не всегда установки по умолчанию приводят к правильным результатам. Приведем пример [8], иллюстрирующий, что может получиться, если при построении графика сильно осциллирующей функции использовать число узлов по умолчанию (первый рисунок), и как изменится картинка при увеличении числа узлов (второй рисунок).

```
> plot((x-25)^2/10+cos(2*Pi*x),x=0..49);
```




```
> plot((x-25)^2/10+cos(2*Pi*x),x=0..49,numpoints=500);
```



Команда plot

Основной и наиболее часто используемой командой двумерной графики является команда plot из ядра Maple. В зависимости от входных параметров она позволяет рисовать графики функций одной переменной, параметрически заданных функций, наборов точек и т. д.

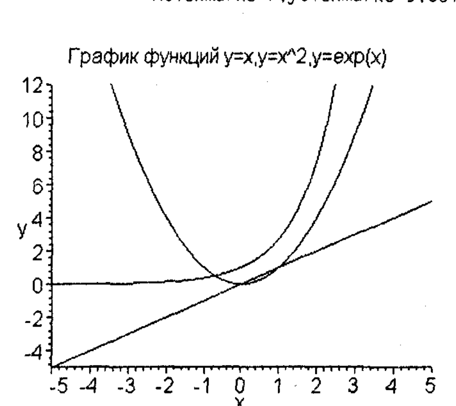
Выше было показано, как применять эту команду для вывода графика функции одной действительной переменной. Приведем вариант команды plot, который позволяет в одних осях строить графики нескольких функций. Отметим, что аналогично можно использовать все команды двумерной графики, то есть в качестве входного параметра можно использовать множество функций, наборов точек и т. д. В этом случае обращение к команде имеет вид:

```
plot({func1,func2...},x=a..b,y=c..d,<options>);
```

Здесь func1, func2, ... — выражения, зависящие от переменной x, a..b — диапазон изменения переменной x (отрезок оси абсцисс), c..d — выводимый интервал по оси ординат. Здесь и далее в угловых скобках указаны аргументы, которые могут отсутствовать. Если одним из концов интервала изменения x является бесконечность, то выводится график асимптотического поведения функции. Приведем

пример построения графиков трех функций и используем параметры для явного указания числа насечек по осям координат, вида осей координат, цвета линий и заголовка:

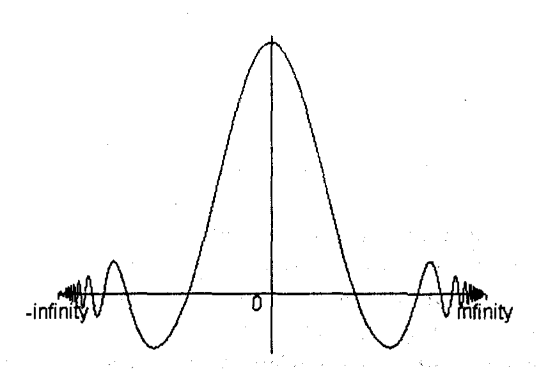
```
> plot({x,x^2,exp(x)},x=-5..5,y=-5..12, axes=FRAME,
                                   title="График функций y=x,y=x^2,y=exp(x)",
                                   xtickmarks=7,ytickmarks=9,color=black);
```



Вообще говоря, аргумент функции может отсутствовать. В этом случае в качестве параметра указывается имя функции, а при задании интервала изменения аргумента не используется идентификатор переменной.

Приведем пример, включающий описание функции f и вывод ее графика на бесконечном интервале:

```
> f:=x->sin(x)/x:
> plot(f,-infinity..infinity,numpoints=300);
```

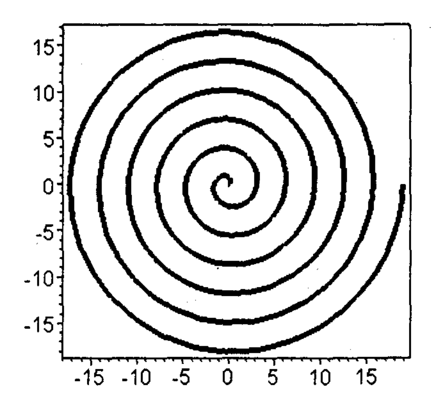


Команду plot можно применять и для вывода параметрически заданной кривой:

```
plot([funx(t),funy(t),t=a..b],<options>),
```

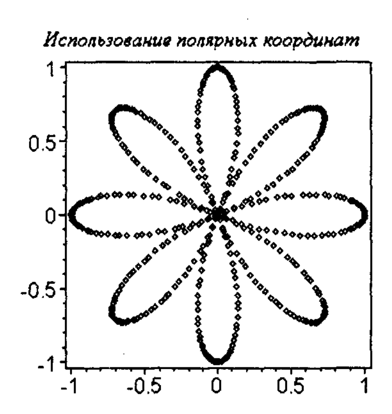
Здесь funx(t), funy(t) — функции координат, зависящие от параметра t; a и b — границы интервала изменения параметра. Например:

```
> plot([t*cos(2*t),t*sin(2*t),t=0..6*Pi],
  scaling=CONSTRAINED,axes=boxed,thickness=4);
```



По умолчанию график строится в декартовых координатах, однако можно определить и другие типы координат. В следующем примере мы используем полярные координаты:

```
> plot( cos(4*t),t=0..6*Pi, numpoints=200,
  title ="Использование полярных координат", axes= boxed,
  style=point, symbol=diamond, symbolsize=10, coords=polar,
  scaling=CONSTRAINED,color=black,
  titlefont=[TIMES,ITALIC,10]);
```



Отметим, что аналогичного результата можно добиться с помощью команды polarplot. В примере использованы следующие параметры: numpoints — для указания числа точек при построении кривой, title — для введения заголовка, axes — для определения типа осей координат, style, symbol и symbolsize — для задания вида маркера при изображении кривой точками, coords — для указания полярной системы координат, scaling — для изображения с одинаковым масштабом по осям, titlefont — для определения шрифта заголовка.

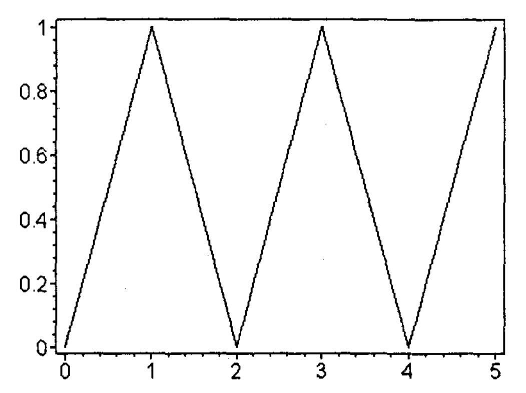
Команда plot может использоваться и для изображения набора точек:

```
plot([[x1,y1],[x2,y2],...],x=a..b,y=c..d,<options>);
```

Здесь [[x1,y1],[x2,y2],...] — набор точек, x1, x2, ... — абсциссы, а y1, y2, ... — ординаты. **Обязательным параметром в случае вывода набора точек является сам набор.**

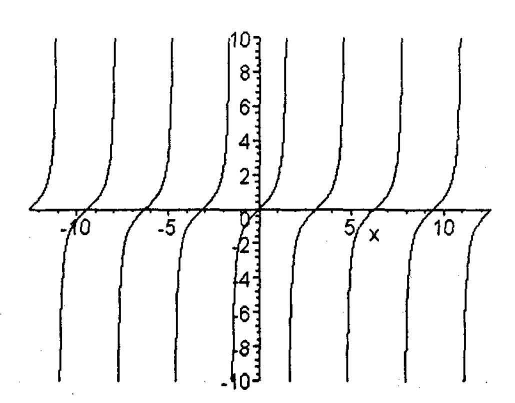
По умолчанию точки соединяются линиями. Приведем пример, в котором при помощи команды plot выводится набор точек с именем dat:

```
> dat:=[[0,0],[1,1],[2,0],[3,1],[4,0],[5,1]]:
> plot(dat,axes=BOXED);
```



Отметим, что команда plot предоставляет и другие возможности для построения графиков. В частности, с использованием параметра discont можно корректно выводить графики разрывных функций. Например, если просто дать команду рисования графика тангенса, то получится набор пиков в точках разрыва, а применение параметра discont позволяет получить корректный график:

```
> plot(tan(x),x=-4*Pi..4*Pi,-10..10,discont=true);
```



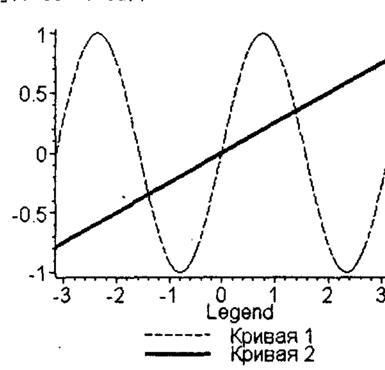
Специальные команды двумерной графики

Описание команд специализированных двумерных графических построений начнем со служебных команд. Очень важной такой командой является display. Часто бывает нужно совместить на одном рисунке графические образы, полученные при помощи различных графических команд. Для этого результат действия каждой команды должен быть присвоен некоторой переменной (при этом вывода на экран не происходит), а затем нужно обратиться к команде

```
display([str1,str2,...],<options>);
```

Здесь в квадратных скобках стоит список (или массив), элементами которого являются графические структуры str1, str2, ... (полученные в результате действия двумерных графических команд), а в качестве параметров могут фигурировать все описанные ранее. В результате выполнения команды display все рисунки, соответствующие графическим структурам, будут выведены в одних осях координат. Проиллюстрируем действие этой команды на примере совмещения графиков двух функций с разными типами линий:

```
> pic1:=plot(sin(2*x),x=-Pi..Pi,linestyle=3,
   legend=["Кривая 1"]):
> pic2:=plot(x/4,x=-Pi..Pi,thickness=3,legend=["Кривая 2"]):
> plots[display]([pic1,pic2],axes=boxed);
```



Команду display с применением параметра insequence можно использовать для создания мультфильмов. По умолчанию значение этого параметра принято false. Если insequence=true, то графические структуры выводятся в поле вывода последовательно, создавая кадры мультфильма.

Для вывода графической структуры с изменением параметров можно использовать следующую команду:

```
replot (GS,<options>)
```

Кроме многофункциональной команды plot в пакете имеются также команды для рисования графиков функции одной переменной:

- logplot(expr,var1=a..b,<options>) — построение графика выражения expr в логарифмическом масштабе по оси y;
- semilogplot(expr,var1=a..b,<options>) — построение графика выражения expr в логарифмическом масштабе по оси x;
- loglogplot(expr,var1=a..b,<options>) — построение графика выражения expr в логарифмическом масштабе как по оси x, так и по оси y;
- polarplot([rad,ang,var=a..b],<options>) — вывод графика в полярных координатах, где функции радиуса rad и угла ang зависят от переменной var, изменяющейся на отрезке [a,b].

В Maple предусмотрена команда перевода графической структуры из одних координат в другие:

```
changecoords (GS,coord)
```

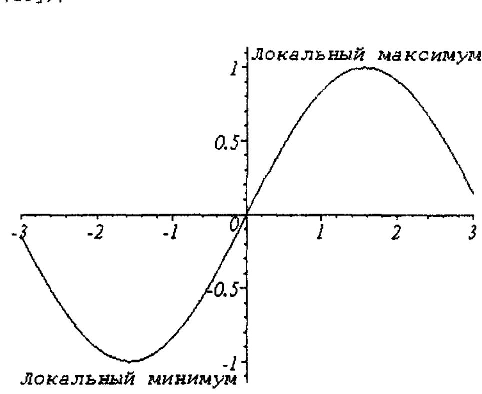
Здесь GS — графическая структура, coord — новая система координат (полярная, декартова и др., см. справку Maple).

Для размещения текста на графике существует команда

```
textplot ([exprx,expry,string],<options>)
```

Результатом ее выполнения будет размещение текстовой строки string начиная с точки с координатами exprx, exprу. Приведем пример:

```
> pic1:=plot(sin,-3..3,color=black):
> pic2:=textplot([-Pi/2,-1.1,"Локальный минимум"]):
> pic3:=textplot([Pi/2,1.1,"Локальный максимум"]):
> display([pic1,pic2,pic3], font=[COURIER,OBLIQUE,10],
 axesfont=[TIMES,ITALIC,10]);
```



Кроме уже описанного в предыдущем разделе способа выводить при помощи команды plot графически точки, заданные своими координатами, существует также возможность использовать следующие команды:

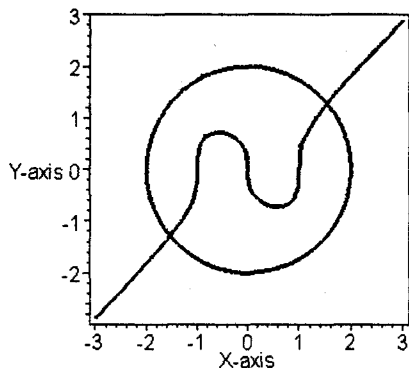
- listplot(L, <options>) — выводит точки из листа данных L. Множество точек L может быть представлено в двух вариантах: [y1,y2,...] и [[x1,y1],[x2,y2],...]. В первом случае в качестве координат по оси x принимаются числа 1, 2, ...;
- polygonplot([[x1,y1],[x2,y2],...],<options>) — построение многоугольника, заданного вершинами [x1,y1],[x2,y2],..., причем последняя точка соединяется с первой;
- pointplot([[x1,y1],[x2,y2],...],<options>) — выводит точки с координатами [x1,y1],[x2,y2], ... Отличие от команды plot состоит в том, что по умолчанию точки не соединяются.

Очень важной и часто используемой является команда рисования графика функции двух переменных, заданной неявно. Эта команда предназначена для изображения линии уровня g функции f(var1,var2) в прямоугольнике [a,b]*[c,d] и имеет вид:

```
implicitplot(f=g, var1=a..b, var2=c..d, <options>)
```

Приведем пример построения двух линий уровня, используя для большей детализации параметр grid. Параметр labels позволяет снабдить оси координат надписями:

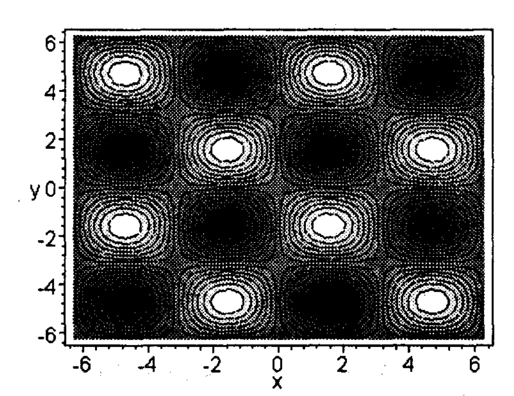
```
> implicitplot({x^3-y^3=x,x^2+y^2=4}, x=-3..3,
y=-3..3,axes=BOXED,grid=[70,70],scaling=constrained,
labels=["X-axis","Y-axis"], thickness=3);
```



Для графического анализа функции двух действительных переменных существует несколько команд. Перечислим эти команды, иллюстрируя их действие примерами:

- contourplot(expr1,x=a..b,y=c..d) — построение линий уровня функции двух переменных x и y, заданной выражением expr1; переменные x и y изменяются на отрезках [a,b] и [c,d] соответственно. У этой команды существует ряд специфических параметров. Параметр contours=n указывает число линий уровня. Можно явно указывать в квадратных скобках значения функции, для которых будут строиться линии уровня. Параметр filled=true/false определяет, закрашивать или нет области между линиями уровня. Цвет закраски задается параметром coloring=[color1,color2] (цвет будет плавно изменяться от color1 до color2). Например:

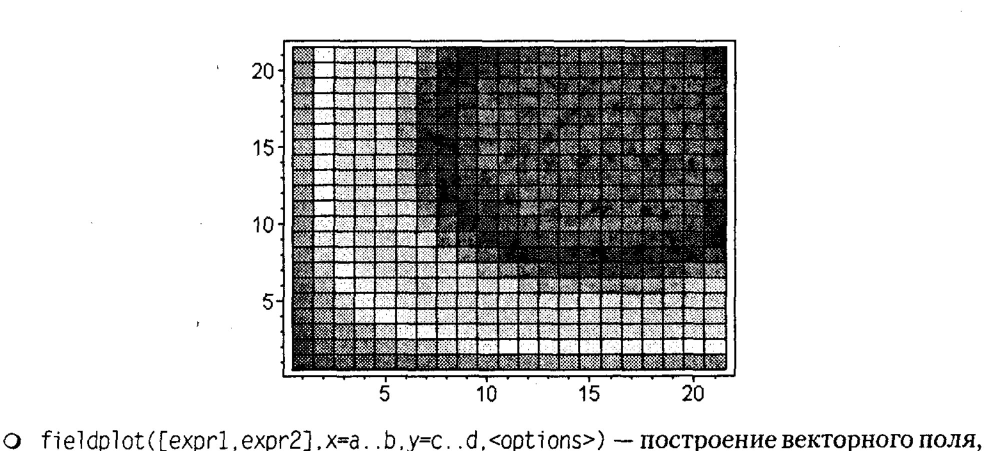
```
> contourplot(sin(x)*sin(y), x=-2*Pi..2*Pi,y=-2*Pi..2*Pi,
   grid=[40,40],contours=15,filled=true,
   coloring=[white,black],axes=boxed);
```



- listcontplot(G, options) – рисование линий уровня функции двух переменных x и y, заданной набором числовых значений G. Здесь G — переменная типа listlist со значениями функции в узлах прямоугольной сетки;
- densityplot(expr,var1=a..b,var2=c..d,<options>) — рисование функции плотности линий уровня для выражения expr, зависящего от переменных var1 и var2;
- listdensityplot(G,var1=a..b,var2=c..d,<options>) — рисование функции плотности линий уровня для набора чисел G. Здесь G — переменная типа listlist, которая задает значение функции в узлах;

Пример:

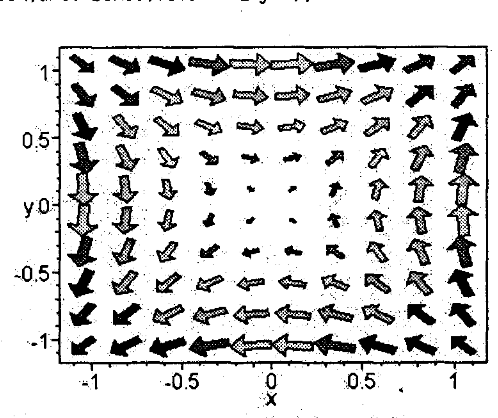
```
> points := [seq( [seq( sin(j*0.1)*sin(i*0.1),
 i=1..21)],j=1..21)]:
> listdensityplot( points, colorstyle=HUE,axes=boxed);
```



- fieldplot([exprl,expr2],x=a..b,y=c..d,<options>) — построение векторного поля, определяемого выражениями exprl и expr2; переменные x и y изменяются на отрезках [a,b] и [c,d] соответственно;
- gradplot(expr,var1=a..b,var2=c..d,<options>) — изображение векторного поля, задаваемого градиентом выражения expr, вычисляемого по переменным var1, var2.

У последних двух команд также существует специфический параметр arrow, указывающий тип выводимых стрелок. Возможны следующие значения: LINE, THIN, SLIM и THICK. Приведем пример:

```
> gradplot(sin(x)*sin(y),x=-Pi/3..Pi/3,y=-Pi/3..Pi/3,
grid=[10,10],arrows=THICK,axes=boxed,color=x^2+y^2);
```

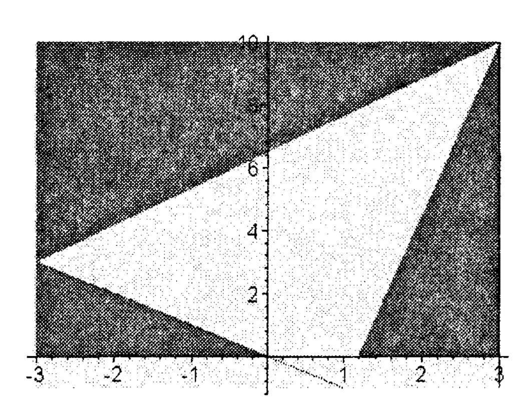


Для графического анализа нескольких линейных неравенств предназначена команда

```
inequal (exprs,var1=a..b,var2=c..d,<options>)
```

Здесь под exprs понимается множество неравенств. Пример:

```
> inequal({x+y<10,y>1,x<=3},x=-3..3,y=-1..10,
 optionsexcluded=(color=yellow),
 optionsfeasible=(color=red));
```

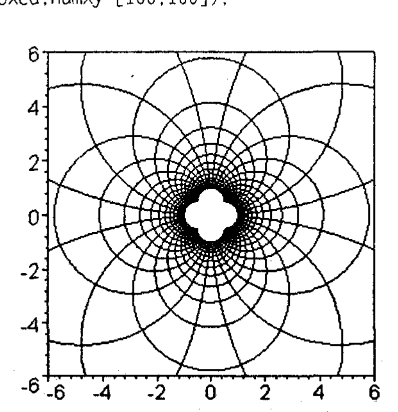


В этом примере в качестве параметров указаны цвета различных областей (удовлетворяющих неравенствам, им противоречащих и др.). Подробнее о параметрах этой команды смотрите в справке Maple.

До сих пор мы рассматривали графические команды, работающие с действительными числами. В пакете есть несколько команд для анализа комплекснозначных функций. Перечислим их:

- `rootlocus(f(s), s, r1..r2,<options>)` — эта команда изображает комплексные корни уравнения `1 + k f(s) = 0`, где параметр `k` принимает значения из интервала `r1..r2`;
- `complexplot(F,p,<options>)` — график кривой, заданной комплекснозначной функцией `F`;
- `conformal(F,r1,r2,<options>)` — изображение конформного отображения комплекснозначной функции `F`. У команды есть дополнительный параметр `numxy=[nx,ny]`, который указывает количество точек на каждой линии уровня по осям координат. Приведем пример:

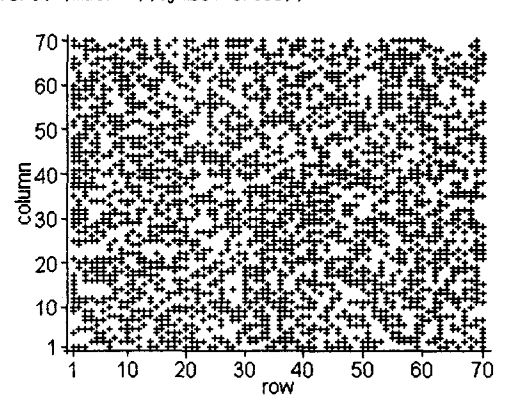
```
> conformal(1/z,z=-1-I..1+I,-6-6*I..6+6*I,grid=[30,30],
 scaling=constrained,axes=boxed,numxy=[100,100]);
```



Следующая команда полезна для визуализации матриц, возникающих в численных приложениях (методы конечных разностей и конечных элементов):

sparsematrixplot(A,<options>) — изображение ненулевых элементов матрицы A. Пример:

```
> A:=LinearAlgebra[RandomMatrix](70,70,generator=0..1):
> sparsematrixplot(convert(A,matrix),symbol=cross);
```

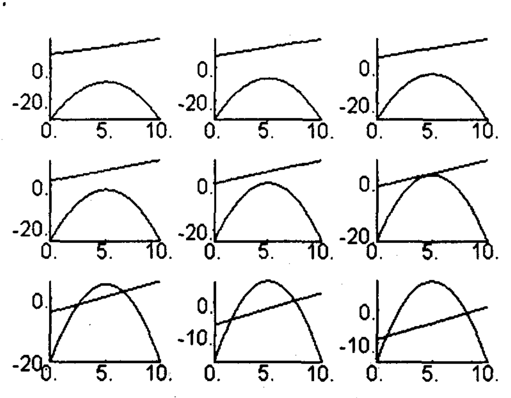


Закончим описание двумерной графики командой двумерной мультипликации:

animate(F,x=a..b,t=c..d,frame=n) — здесь F — выражение, зависящее от переменных x и t, которые изменяются соответственно в интервалах [a,b] и [c,d]. Переменная x отвечает оси абсцисс, t — переменная времени, а параметр frame задает число кадров.

Приведем пример обращения к этой команде для рисования графиков прямой и параболы, зависящих от параметра. В результате на экране возникает окно для просмотра мультфильма. Приведем девять кадров:

```
> animate( {x-t,-(x-5)^2+t},x=0..10,t=-9..9,
 frames=40,axes=boxed);
```



Трехмерная графика

Организация работы команд трехмерной графики та же, что и двумерной графики. Многие команды трехмерной графики аналогичны командам, рассмотренным

в предыдущем разделе, и отличаются тем, что их имена оканчиваются на 3d. При этом число параметров, как правило, больше на единицу, а точка задается тремя координатами. Потому структура этого раздела совпадает со структурой предыдущего: сначала идет описание графических структур, затем команд из пакета plottools, потом графических параметров, команды plot3d и в последнем разделе перечислены команды для специальных трехмерных математических построений.

Отметим, что в предыдущих версиях Maple для изображения трехмерных графических структур применялась команда display3d, отличающаяся от команды display. В версии 6.0 она осталась, но ее функции также выполняет команда display.

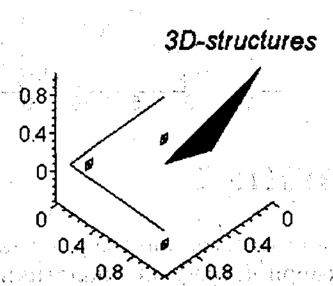
Трехмерные графические структуры

Создание трехмерных рисунков, так же как и двумерных, происходит в три этапа. На первом этапе графическая команда производит необходимые вычисления и представляет их результат в виде набора трехмерных объектов. На втором этапе эти объекты преобразуются в графическую структуру командой PLOT3D (обращение к ней аналогично обращению к команде PLOT), а затем выводятся на экран или другое устройство, например в файл, или присваиваются переменной. Графическая структура может содержать параметры, которые управляют представлением рисунка.

Трехмерные графические объекты POINTS, CURVES, POLYGONS, TEXT аналогичны соответствующим двумерным объектам и имеют на одну размерность больше при задании координат точек. То же относится и к большинству параметров трехмерных графических структур, поэтому не будем их здесь описывать, а ограничимся примерами.

Сформируем четыре трехмерные графические структуры: первая описывает кривую, проведенную через три точки, вторая — набор точек, третья — текст, а четвертая — треугольник. При обращении к команде рисования PLOT3D используем параметры, задающие цвет, размер и тип маркера, масштабирование рисунка и вид координатных осей:

```
> str1:=CURVES([[0,0,0],[1,0,0],[1,1,0]]):
> str2:=POINTS([[0.1,0.1,-0.3],[0.9,0.1,0],[1.1,1.1,0]]):
> str3:=TEXT([0,0,0.5],`"3D-structures"`,
   FONT(HELVETICA,OBLIQUE,12)):
> str4:=POLYGONS([[0.5,0.5,0],[0.5,1.0,0.5],[0,0,1,1]]):
> PLOT3D(str1,str2,str3,str4,
   COLOR(RGB,0,0,0),SYMBOL(DIAMOND,25),
   SCALING(CONSTRAINED),AXESSTYLE(FRAME));
```



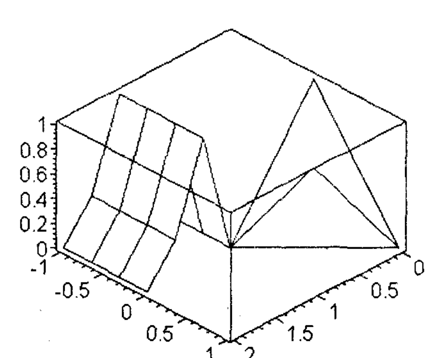
Опишем специфические трехмерные графические объекты:

- GRID(a..b,c..d,Z) — задание поверхности над участком координатной плоскости [a,b]*[c,d] при помощи переменной типа listlist, которая имеет вид Z=[[z11,...z1n],..., [zm1...zmn]]. Размерности этой переменной n и m определяют число равноотстоящих узлов по осям x и y соответственно. Каждый элемент переменной Z задает z-координату в соответствующем узле;
- MESH(A) — задание поверхности при помощи переменной A типа listlist, которая имеет вид A=[[[x11,y11,z11],...,[x1n,y1n,z1n]],...,[[xm1,ym1,zm1],...,[xmn,ymn,zmn]]]. В отличие от предыдущей команды здесь указываются все три координаты точек поверхности, что позволяет использовать неравномерную сетку.

Перед тем как перейти к заключительному примеру использования трехмерных графических объектов, перечислим некоторые специфические параметры команды PLOT3D (трехмерной графической структуры): GRIDSTYLE задает способ соединения узлов поверхностей и может принимать значения TRIANGULAR и RECTANGULAR, а вид подсветки поверхности задается параметрами LIGHT и LIGHTMODEL. Кроме того, параметр раскраски COLOR в трехмерном случае может принимать дополнительные значения XYZSHADING и XYSHADING, параметр STYLE при значении CONTOUR рисует сетку, а при значении PATCHCONTOUR еще и закрашивает поверхность.

Теперь разными командами создадим два графических объекта (поверхности) и нарисуем их:

```
> str1:=GRID(1..2,-1..0,[[0,0,0,0],[1,1,1,1],
[0.3,0.3,0.3,0.3], [0,0,0,0]]):
> str2:=MESH([[[1,1,1],[0,1,0],[0,0,1]],
   [[1,1,1],[1,0,0],[0,0,1]]]):
> PLOT3D(str1,str2,AXESSTYLE(BOXED),COLOR(XYSHADING));
```



Трехмерные команды пакета plottools

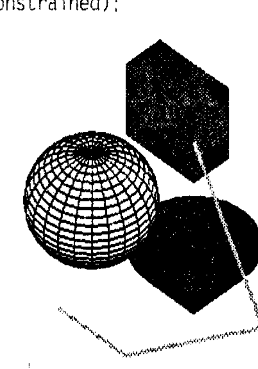
Работа с графическими структурами с помощью команд пакета plottools в трехмерном случае во многом аналогична двумерному. Команды arrow, polygon, point и некоторые другие сохраняются, но при обращении к ним нужно добавлять еще одну координату. У команд, оперирующих с графическими структурами (reflect, rotate, scale, translate, project), возникает дополнительный параметр, отвечающий третьей размерности. Эти команды мы в данном разделе не описываем, а только ис-

пользуем в примерах. Однако существуют и специфические трехмерные команды, полный список которых можно найти в справке по пакету plottools. Ниже мы перечислим только некоторые из них:

- cone([x,y,z], r, h) — формирует усеченный конус с началом в точке с координатами [x,y,z]. Необязательные параметры r и h определяют соответственно высоту и диаметр основания. По умолчанию они принимаются равными 1;
- cuboid([x1,y1,z1], [x2,y2,z2]) — задает параллелепипед с диагональю, определяемой точками [x1,y1,z1] и [x2,y2,z2];
- curve([[x1,y1,z1],[x2,y2,z2],...]) — определяет пространственную ломаную, проходящую через точки с координатами [x1,y1,z1], [x2,y2,z2], ...;
- cylinder(c, r, h) — формирует цилиндр с центром основания в точке с координатами c=[x,y,z]. Необязательные параметры r и h определяют соответственно радиус и высоту цилиндра. По умолчанию они принимаются равными 1;
- hemisphere([x,y,z], r) — задает полусферу с центром в точке [x,y,z] и радиусом r. По умолчанию радиус r равен 1;
- sphere([x,y,z], r) — определяет сферу с центром в точке [x,y,z] и радиусом r. По умолчанию радиус r равен 1;
- torus([x,y,z], r, R) — задает тор с центром в точке [x,y,z], радиусом r и расстоянием R между центром меридиана и тора. По умолчанию радиус r равен 1, а R равно 2.

Напомним, что перечисленные команды только формируют графические объекты, а для их изображения нужно использовать команду display (или display3d). Приведем пример, демонстрирующий действие трехмерных команд пакета plottools. Сформируем графический объект d из конуса и сферы, а в качестве объектов f и l определим параллелепипед и ломаную соответственно:

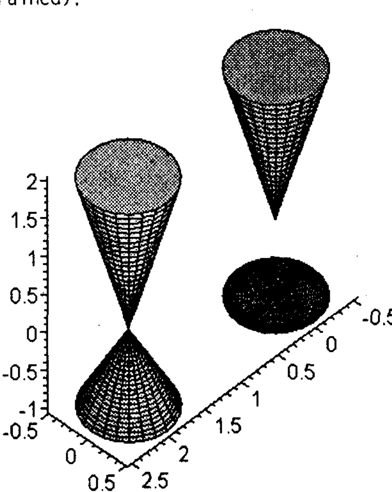
```
> d := cone([-1,0,0,-1.5],0.7,color=red),
  sphere([0,-0.5,0.5],0.7,color=yellow):
> f := cuboid([-2.5,-2,-1],[-2,-1,0]):
  l := curve([[-1,0,1],[-1,1,-1],[1,1,0],[1,0,0]],
color=green, thickness=3):
  plots[display](d,f,l,scaling=constrained);
```



Следующий пример иллюстрирует работу команд, оперирующих с трехмерными графическими структурами. Вначале сформируем графическую структуру в виде

конуса, затем повернем его, перенесем по горизонтальной координате, отразим с уменьшением и спроецируем его на плоскость:

```
> s:=plots[display](cone([0,0,0],0.5,2)):
> s1:=rotate(s,0,0,Pi/2): s2:=translate(s,2,0,0):
> s3:=scale(reflect(s,[1,0,0]),1,1,1/2):
> s4:=project(s,[[0,0,-1],[1,-1,-1],[2.5,0,-1]]):
> plots[display](s,s1,s2,s3,s4,
  axes=framed,scaling=constrained);
```



Управляющие параметры трехмерной графики

В командах трехмерной графики для управления видом рисунка используются параметры, частично совпадающие с параметрами двумерной графики. Коротко перечислим основные из них, не останавливаясь подробно на уже описанных ранее:

- coords=opt — тип используемой системы координат (CARTESIAN, SPHERICAL или CYLINDRICAL);
- title="text" — заголовок, содержащийся в строке "text";
- axes=opt — тип осей координат (FRAME, NORMAL, BOXED, NONE);
- scaling=opt — тип масштабирования (UNCONSTRAINED, CONSTRAINED);
- orientation=[angle1,angle2] — ракурс; углы angle1, angle2 даются в градусах;
- view=az..bz или view=[ax..bx,ay..by,az..bz] — выводимая на рисунок область (все за ее пределами отсекается);
- projection=n — тип проекции (перспектива), 0≤n≤1. Так называемый рыбий глаз при n=0, а при n=1 соответствует проекции без учета перспективы;
- style=opt — стиль вывода (POINT — точки, LINE — линии, HIDDEN — сетка с удалением невидимых линий, PATCH — заполнитель, WIREFRAME — сетка с выводом невидимых линий, CONTOUR — линии уровня, PATCHCONTOUR — заполнитель и линии уровня);

- contours=n — число линий уровня функции двух переменных;
- shading=opt — задание функции интенсивности заполнителя; параметр opt может принимать следующие значения: Z (функция координаты Z), XY (функция координат X и Y), XYZ (функция трех координат), ZGREYSCALE, ZHUE, NONE (без раскраски);
- grid=[m,n] — число узлов по осям x и y для вычисления поверхности (по умолчанию grid=[25,25]);
- gridstyle=opt — тип сетки (triangular/rectangular — треугольная/прямоугольная);
- numpoints=n — минимальное число узлов для вычисления поверхности; этот параметр эквивалентен параметру grid;
- color=opt — цвет поверхности (red,blue,...). В качестве opt может фигурировать конструкция, аналогичная заданию цвета в графических структурах, а также функция;
- light=[an1,an2,r,g,b] — задание подсветки, создаваемой источником света из точки со сферическими координатами an1, an2. Цвет подсветки определяется долями красного (r), зеленого (g) и синего (b) цветов, которые находятся в интервале от 0 до 1;
- ambientlight=[r,g,b] — цвет изображения, определяемый долями красного (r), зеленого (g) и синего (b) цветов, которые находятся в интервале от 0 до 1;
- tickmarks=[i1, i2, i3] — задание числа насечек по осям координат;
- labels=[str1,str2,str3] — надписи по осям координат, задаваемые строками str1, str2, str3;
- filled=true/false — закрашивание пространства между поверхностью и плоскостью (X,Y) в случае значения true. Этот параметр применим только к командам plot3d, contourplot3d и listcontplot3d.

Пользователь может переопределять установки по умолчанию на сеанс при помощи команды

```
setoptions3d (opt1=val1,..., optn=valn)
```

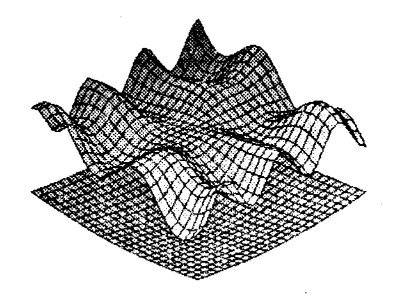
Команда plot3d

Подобно двумерной графике, где функция plot выполняет самые разнообразные функции, в трехмерном случае существует многофункциональная команда

```
plot3d({ex1,ex2,...},var1=a..b,var2=c..d,<options>);
```

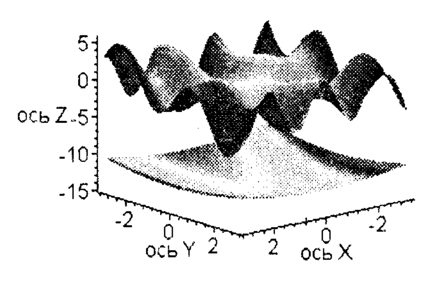
Здесь приведен один из видов обращения к этой команде, позволяющий выводить на одном рисунке несколько поверхностей, задаваемых однотипными Maple-выражениями ex1, ex2, ... Эти выражения могут зависеть только от двух переменных var1 и var2. Поверхности рисуются для прямоугольника a≤var1≤b, c≤var2≤d. Вид представления рисунка можно менять при помощи параметров (options). Приведем пример использования этой команды для изображения двух поверхностей. Для иллюстрации действия различных параметров выполним команду два раза — без использования и с использованием параметров:

```
> plot3d({x*sin(2*y)+y*cos(3*x),sqrt(x^2+y^2)-15},
x=-Pi..Pi,y=-Pi..Pi);
```



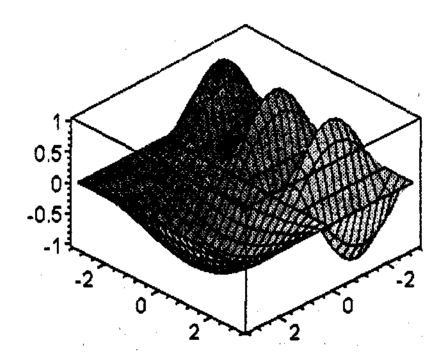
```
> plot3d({x*sin(2*y)+y*cos(3*x),sqrt(x^2+y^2)-15},
  x=-Pi..Pi,y=-Pi..Pi, grid=[40,40], style=PATCHNOGRID,
  title="Команда plot3d", light=[1,1,1,1,1], axes=FRAMED,
  labels=["ось X","ось Y","ось Z"], orientation=[50,70],
  color=x^2+y,titlefont=[TIMES,BOLD,14]);
```

Команда plot3d



Как и для команды plot, аргументы функции не обязательны. Если они отсутствуют, то в качестве первого параметра команды plot3d указывается только имя функции, после которого следует пара интервалов изменения переменных. Проиллюстрируем это примером, в котором опишем функцию двух переменных и изобразим соответствующую ей поверхность:

```
> fun:=proc(x,y) local m; m:=0;
  if x<0 then m:=sin(x)*sin(3*y) else m:=-sin(x)*cos(y/2)
  fi;  m; end:
> plot3d(fun,-Pi..Pi,-Pi..Pi,axes=BOXED);
```





ВНИМАНИЕ При графическом изображении кусочно-непрерывных или вычисляемых функций, например функции fun из предыдущего примера, могут возникать трудности. Так, будет получено сообщение об ошибке, если попытаться обратиться к команде plot3d следующим образом:

```
> plot3d(fun(x,y),x=-Pi..Pi,y=-Pi..Pi);
Error, (in fun) cannot evaluate boolean: x < 0
```

Это объясняется тем, что Maple сначала обращается к функции, исходя из того что переменная x является символьной, а затем подставляет числовые значения. От этой ошибки можно избавиться, заключив в кавычки обращение к функции. Например, следующая команда даст правильный результат:

```
> plot3d("fun(x,y)",x=-Pi..Pi,y=-Pi..Pi);
```

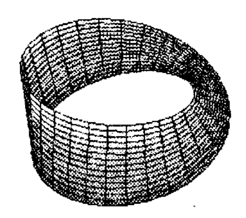
Аналогично команде plot команда plot3d предназначена для вывода заданной параметрически поверхности. В этом случае обращение к команде имеет вид:

```
plot3d([expr1,expr2,expr3],var1=a..b,var2=c..d)
```

Здесь *координаты заданы выражениями* expr1, expr2, expr3 *от двух переменных* var1 и var2, которые изменяются на отрезках [a,b] и [c,d] соответственно. Пример:

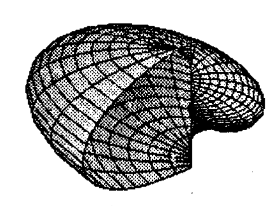
```
> plot3d([(5+cos(t/2)*u)*cos(t),
  (5+cos(t/2)*u)*sin(t),sin(t/2)*u], t=0..2*Pi,
   u=-1..1, grid=[30,15], orientation =[-130,30],
   titlefont=[HELVETICA,BOLD,20], title="Лист Мебиуса.",
   shading=ZGREYSCALE);
```

Лист Мебиуса.



При помощи команды plot3d можно осуществлять и некоторые другие операции. Например, используя параметр coords, можно строить поверхности в сферических и цилиндрических координатах:

```
> S:=plot3d([f,f,g],f=0..2*Pi,g=0..Pi,coords=spherical): S;
```

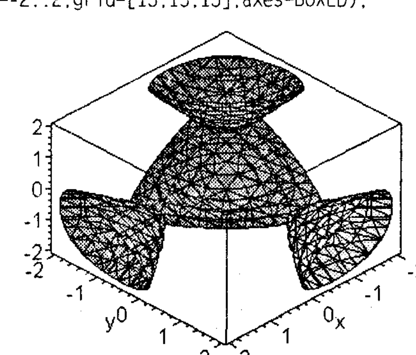


Специальные команды трехмерной графики

В данном разделе приводится ряд наиболее популярных команд трехмерной графики, а их полный список можно найти в справке Maple:

- contourplot3d(expr,var1=a..b,var2=c..d,<options>) — построение линий уровня выражения expr от двух переменных var1 и var2 в прямоугольнике [a,b]*[c,d] (идентично команде plot3d с параметром style=CONTOUR);
- cylinderplot(expr,var1=a..b,var2=c..d,<options>) — изображение поверхности, определяемой выражением expr, заданным в цилиндрических координатах;
- fieldplot3d([expr1,expr2,expr3], var1=a..b, var2=c..d, var3=e..f, <options>) — построение в параллелепипеде [a,b]*[c,d]*[e,f] трехмерного векторного поля с компонентами, заданными выражениями expr1, expr2, expr3, которые зависят от переменных var1, var2, var3;
- gradplot3d(expr,var1=a..b,var2=c..d,var3=e..f,<options>) — построение трехмерного поля градиента выражения expr, зависящего от трех переменных var1, var2, var3. Переменные изменяются в заданных интервалах;
- implicitplot3d(expr=g,var1=a..b,var2=c..d,var3=e..f, <options>) — построение неявно заданной поверхности expr=g в параллепипеде [a,b]*[c,d]*[e,f]. Здесь выражение expr зависит от переменных var1, var2, var3. Дадим пример использования последней команды:

```
> implicitplot3d( x^3 + y^3 + z^3 = (x + y + z + 1)^3,
   x=-2..2,y=-2..2,z=-2..2,grid=[15,15,15],axes=BOXED);
```



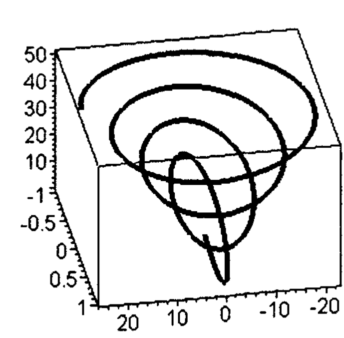
- matrixplot(M,<options>) — построение поверхности, заданной матрицей M. При задании параметра heights=histogram строится трехмерная гистограмма. Например:

```
> matrixplot([[-1,3,4],[-2,-1,0],[3,4,8]],heights=histogram,
  shading=zgrayscale, axes=box,orientation=[-75,75],
  tickmarks=[[1.5="январь",2.5="февраль",3.5="март"],
  [1.5=1998,2.5=1999,3,5=2000],5],labels=["","","T"]);
```

- odeplot(s,vars,r1,r2,<options>) — изображение решения дифференциального уравнения (см. примеры в главе 4 «Решение уравнений в Maple»);
- pointplot([[x1,y1,z1],...,[xn,yn,zn]],<options>) — вывод точек, заданных своими координатами;
- polygonplot3d([pnt1,...,pntn]) — построение трехмерного n-угольника, заданного точками pnt1, ..., pntn, причем первая точка соединяется с последней;
- polyhedraplot(L,<options>) — изображение набора точек в виде трехмерных геометрических фигур. Здесь L — переменная типа listlist, в которой находятся координаты точек. Каждая точка является центром изображаемой фигуры. Параметр polyscale=n задает размер фигуры (по умолчанию n=1), а параметр polytype=p определяет вид фигуры, где p может принимать следующие значения: tetrahedron, octahedron, hexahedron, dodecahedron, icosahedron;
- spacecurve([expx,expy,expz],var=a..b,<options>) — построение кривой в трехмерном пространстве. Кривая задается параметрически выражениями координат expx, expy и expz, зависящими от одной переменной var, изменяющейся на интервале [a,b].

  Пример:

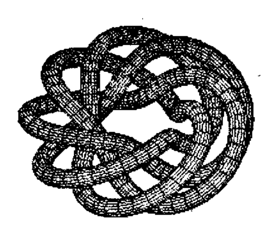
```
> spacecurve({[t*cos(t),sin(t),2*t]},t=0..8*Pi,
  numpoints=200,axes=BOXED,
  orientation=[80,50],color=black,thickness=4);
```



- sphereplot(expr,var1=a..b,var2=c..d,<options>) — построение поверхности, заданной в сферических координатах. Выражение expr задает радиус, зависящий от двух угловых переменных var1 и var2;
- surfdata([[[x11,y11,z11],...,[x1n,y1n,z1n]],...,[[xm1,ym1,zm1]...[xmn,ymn,zmn]]], <options>) — построение поверхности, заданной переменной типа listlist;
- textplot3d([expx,expy,expz],str,<options>) — вывод текстовой строки str с центром в позиции, задаваемой координатами expx, expy и expz;
- tubeplot([expx,expy,expz],var=a..b,radius=r,<options>) — построение поверхности вращения, полученной обращением радиуса r вокруг параметрически заданной пространственной кривой. Кривая определяется выражениями для координат expx, expy и expz, зависящими от одной переменной var, изменяющейся на интервале [a,b]. Приведем пример построения сложного тора:

```
> r:=2+0.8*cos(7*t):
```

```
> tubeplot([r*cos(4*t),r*sin(4*t),sin(7*t)],t=0..2*Pi,
 radius=0.25,tubepoints=20,numpoints=200,
 orientation=[45,20]);
```

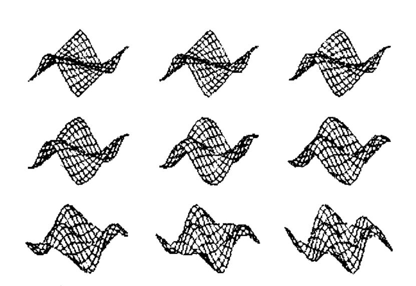


- setoptions3d(options1=exp1,...,optionsn=expn) — назначение параметров трехмерной графики на сеанс.

Коротко коснемся возможностей трехмерной мультипликации. Один способ заключается в использовании команды display3d или display с использованием параметра insequence=true, а другой использует команду animate3d(F, x=a..b, y=c..d, t=t1..t2, frames=n, <options>). Здесь F — выражение, зависящее от переменных x (абсцисса), y (ордината) и t (переменная времени), которые изменяются соответственно в интервалах [a,b], [c,d] и [t1,t2]. Параметр frame задает число кадров.

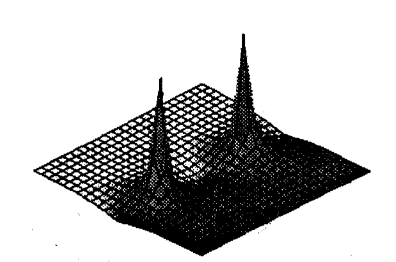
Приведем пример трехмерной анимации, а для создания картинки, состоящей из серии кадров, используем команду display:

```
> res:=animate3d(sin(r*x)*cos(y),x=-Pi..Pi,y=-Pi..Pi,
   r=0.2..1.2,frames=9,color=white,grid=[15,15]):
> display(res);
```



Теперь представим команду для вывода поверхности, определяемой комплекснозначной функцией — complexplot3d(expr, z=a + b*I..c+d*I). Здесь expr — комплекснозначное выражение, зависящее от переменной z. Пример:

```
> complexplot3d( tan(z) , z = -3 - 3*I .. 3 + 3*I );
```

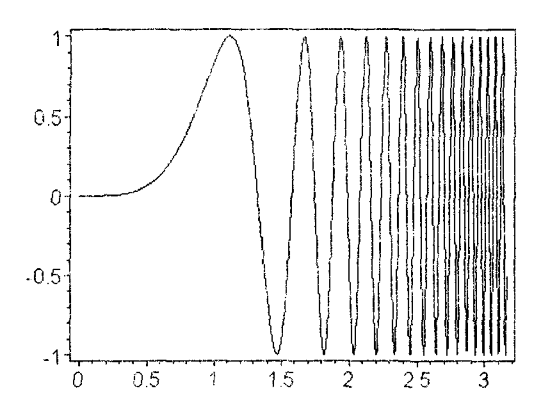


Сложные операции с графикой

В заключение остановимся на возможностях совмещения двумерных и трехмерных графических структур. Напомним, что совмещать однотипные структуры на одной картинке можно с помощью команды display. Опишем ряд команд:

- addcoords(coord_name, v, in_cart, con, dlist) — позволяет вводить собственную систему координат, а затем использовать ее с помощью параметра coords. Здесь coord_name — имя новой системы координат, v — список переменных, in_cart — выражения для декартовых координат в новых переменных, con — список идентификаторов новых координат, dlist — список назначений по умолчанию. Приведем пример использования этой команды для двумерного случая:

```
> addcoords(cc,[e1,e2],[sqrt(e1),sin(e2*e1)]):
> plot(x,x=0..10,coords=cc,axes=boxed,color=black);
```



- changecoords(p,coord) — перевод координат в графической структуре p из текущей системы в систему coord.

Теперь остановимся на команде transform из пакета plottools, которая позволяет преобразовывать графические структуры с помощью задаваемых пользователем отображений (процедур-функций). Обращение к ней имеет вид transform(f), где f - R^m -> R^n.

Объясним работу этой команды на примерах. Создадим двумерную графическую структуру s1, содержащую линии уровня функции sin(x)*cos(y):

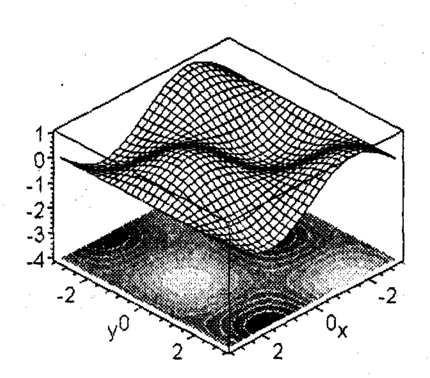
```
> s1:=contourplot(sin(x)*cos(y),x=-Pi..Pi,y=-Pi..Pi,
 filled=true, coloring=[black,white],contours=9,axes=none):
```

Следующей командой переменной s2 присвоим структуру, содержащую поверхность, задаваемую той же функцией:

```
> s2:=plot3d(sin(x)*cos(y),x=-Pi..Pi,y=-Pi..Pi,
  grid=[30,30],style=HIDDEN, color=black):
```

Теперь определим отображение f, которое преобразует двумерные объекты в трехмерные, и выведем на одном рисунке поверхность s2 и преобразованную структуру s1:

```
> f:=transform((x,y)->[x,y,-4]):
> display(s2,f(s1),axes=boxed);
```



Работа с графикой в интерактивном режиме

Многие характеристики графических объектов можно менять интерактивно, обращаясь к соответствующим пунктам меню. При работе в Windows-версии результаты работы графических команд либо вставляются непосредственно в рабочий документ, либо выводятся в отдельном окне, что зависит от установленного в сеансе режима (Inline или Window, см. подпункт Plot Display пункта Options основного меню). При выделении рисунка с помощью мыши в первом случае или при обращении к графической команде во втором случае меню Maple изменяется, и возникают пункты и значки, предназначенные для работы с графическими объектами. Те же самые пункты входят в состав контекстного меню, которое возникает при нажатии правой кнопки мыши в поле рисунка. Отметим, что ряд значков и пунктов основного меню Maple сохраняется.

Набор значков, пунктов и подпунктов зависит от типа графических объектов и различен для двумерных, трехмерных рисунков и анимаций (см. рис. 6.1, 6.2 и 6.3). Названия большинства пунктов графических меню совпадают с именами соответствующих параметров, и при их описании мы ограничимся ссылкой на соответствующий графический параметр. Далее мы коротко коснемся некоторых пунктов и опишем значки графических меню.

Меню двумерной графики

Характерный вид меню двумерной графики дан на рис. 6.1. Информация о пунктах и значках в главном меню сведена в табл. 6.1 и 6.2.

Отметим, что слева от значков в меню двумерной графики находится прямоугольник (см. рис. 6.1), в который выводятся координаты точки, отмечаемой мышью.

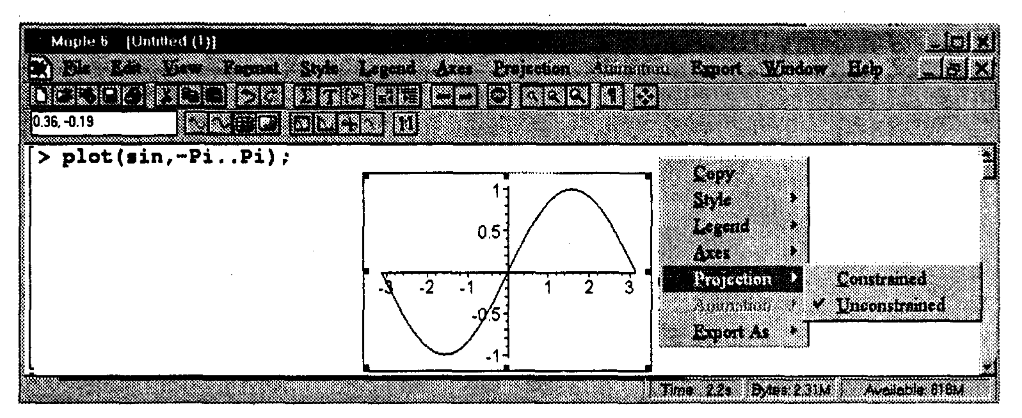


Рис. 6.1. Меню режима интерактивной работы с двумерной графикой

Таблица 6.1. Пункты меню двумерной графики

Пункт меню	Назначение
Style	Управление стилем вывода графика. Изменение типа вывода (линии или точки), типов и размеров символов и линий. Соответствует изменению параметров двумерной графики `style`, `thickness`, `linestyle`, `symbol` и `symbolsize`
Legend	Редактирование комментария к рисунку. Соответствует параметру `legend`
Axes	Управление типом осей координат. Соответствует параметру `axes`
Projection	Изменение типа масштаба рисунка. Соответствует параметру `scaling`
Animation	Этот пункт становится активным только в случае анимаций и предназначен для управления их просмотром. Все пункты меню дублируются значками, которые описаны в табл. 6.5
Export	Запись рисунка на диск в различных графических форматах. Соответствует использованию команды `plotsetup`

Таблица 6.2. Значки меню двумерной графики

Значок	Назначение
	Вывод рисунка линиями (`style=line`)
	Вывод рисунка точками (`style=point`)
	Заполнитель с разделяющими линиями (`style=patch`)
	Заполнитель без разделяющих линий (`style=patchnogrid`)
	Оси координат в виде прямоугольника (`axes=boxed`)
	Оси координат с центром в левом нижнем углу рисунка (`axes=framed`)
	Обычные оси координат (`axes=normal`)
	Рисунок выводится без осей координат (`axes=none`)
	Задание типа масштаба рисунка (параметр `scaling`)

Меню трехмерной графики

Меню трехмерной графики изображено на рис. 6.2, характеристики его пунктов даны в табл. 6.3, а значки описаны в табл. 6.4.

Таблица 6.3. Пункты меню трехмерной графики

Пункт меню	Назначение
Style	Управление стилем вывода графика. Изменение типа вывода (заполнитель, сетка, линии, точки), типов и размеров символов и линий. Соответствует изменению параметров трехмерной графики style, thickness, linestyle, gridstyle, symbol и symbolsize
Color	Управление цветом графических объектов. Соответствует параметрам color, shading и lightmodel
Axes	Управление типом осей координат. Соответствует параметру axes
Projection	Изменение типа масштаба рисунка. Соответствует параметрам projection и scaling
Animation	Этот пункт становится активным только в случае анимаций и предназначен для управления их просмотром. Все пункты меню дублируются значками, которые описаны в табл. 6.5
Export	Запись рисунка на диск в различных графических форматах. Соответствует использованию команды plotsetup

Таблица 6.4. Значки меню трехмерной графики

Значок	Назначение
	При рисовании поверхности используется заполнитель и выводятся соединительные линии. Соответствует параметр style=PATCH
	При рисовании поверхности используется только заполнитель. Соответствует параметру style=PATCHNOGRID
	При рисовании поверхности используется заполнитель и выводятся линии уровня функции. Соответствует параметру style=PATCHCONTOUR
	При рисовании поверхности выводятся только видимые соединительные линии (style=HIDDEN)
	При рисовании поверхности выводятся только линии уровня функции (style=CONTOUR)
	При рисовании поверхности выводятся все соединительные линии (style=WIREFRAME)
	Поверхность выводится точками (style=POINT)
	Оси координат в виде прямоугольника (axes=boxed)
	Обрамляющие оси координат (axes=framed)
	Обычные оси координат (axes=normal)
	Рисунок выводится без осей координат (axes=none)
1:1	Задание типа масштаба рисунка (параметр scaling)

Windows-версия позволяет интерактивно изменять угол проекции трехмерных объектов двумя способами: с помощью изменения значений углов проецирования

(два поля ввода слева от значков трехмерной графики) или путем перемещения указателя при нажатой левой кнопке мыши. Напомним, что углы проецирования можно также указать с помощью параметра orientation.

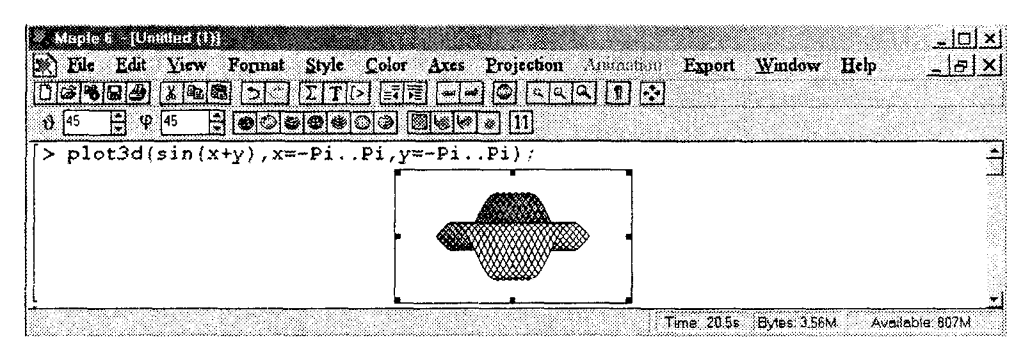


Рис. 6.2. Меню режима интерактивной работы с трехмерной графикой

Рис. 6.3. Меню работы при вызове рисования графика из контекстного меню

Теперь опишем значки, возникающие при обращении к командам анимации animate и animate3d.

Таблица 6.5. Значки меню анимаций

Значок	Назначение
	Остановка анимации
	Старт просмотра анимации
	Переход к следующему кадру

продолжение

Таблица 6.5 (*продолжение*)

Значок	Назначение
	Установка просмотра анимации в обратном порядке
	Установка просмотра анимации в прямом порядке
	Уменьшение скорости просмотра анимации
	Увеличение скорости просмотра анимации
	Установка режима однократного просмотра анимации
	Установка режима повторяющегося в цикле просмотра анимации

В заключение упомянем еще об одной возможности интерактивного использования графики — так называемого «быстрого графика» (`smart plots`). Если установить указатель на математическое выражение, нажать правую кнопку мыши и выбрать пункт Plot в контекстном меню, то будет построен график этого выражения. В этом режиме существует ряд дополнительных возможностей, которые недоступны при обычном вызове графической команды, например можно интерактивно изменять интервалы изменения координат по осям (пункт меню Axes, подпункт Ranges).



Программирование в Maple

О типах переменных, зарезервированных словах, элементарных операциях и стандартных функциях Maple-языка говорилось в предыдущих главах. Здесь речь пойдет о написании пользователем на Maple-языке собственных программ, процедур, отладке программ и др. Для этого в пакете имеется довольно широкий набор команд и конструкций, аналогичный существующим в известных языках программирования (например, в языках Паскаль или С). Ниже мы последовательно перечислим эти конструкции и команды, сопровождая изложение короткими примерами.

Условные операторы

Условный оператор в Maple начинается с зарезервированного слова `if` и обязательно должен заканчиваться словом `fi` или `end if`, которые являются синонимами. Этот оператор имеет несколько форм написания, которые отличаются детальностью проверки условий:

```
if BOOL then EXPR1 fi;
if BOOL then EXPR1 else EXPR2 end if;
```

Эта конструкция дает возможность в зависимости от значения логического условия `BOOL` выполнять выражение `EXPR1` (когда `BOOL=true`) или `EXPR2` (если `BOOL=false` в случае ветки `else`). В качестве выражений `EXPR1` и `EXPR2` здесь может выступать любая последовательность Maple-команд.

Приведем пример:

```
> x:=2: if x<0 then x:=5 else x:=-5 end if;
```

$x := -5$

Для реализации сложных условий можно использовать полный вариант условного оператора, который имеет следующий вид:

```
if   BOOL1 then EXPR1
elif BOOL2 then EXPR2
...
```

```
elif BOOLN then EXPRN
else EXPR0 end if;
```

Таким образом, вложенность условий может быть практически неограниченной и реализуется при помощи вставки elif. В качестве выражений EXPR0, EXPR1, ..., EXPRN могут выступать любые последовательности Maple-команд. Как и для простого условного оператора, у этой конструкции может отсутствовать ветка else. Приведем пример полного условного оператора:

```
> x:=7:
if x<0 then x:=a; elif x=0 then x:=b;
                  elif x<10 then x:=c;
                  else x:=d; fi;
```

$x := c$

Существует вариант условного оператора, который предназначен для использования в выражениях. Его синтаксис имеет вид:

```
"if"(BOOL,EXPR1,EXPR2)
```

При обращении к этому оператору сначала проверяется логическое выражение BOOL, затем при его истинности (true) вычисляется оператор EXPR1, а при ложности (false) — EXPR2.

Пример:

```
> a:=2: b:=4:
> "if"(a>b,log[b](a),log[a](b));
```

$$\frac{\ln(4)}{\ln(2)}$$

```
> simplify(%);
```

2

Операторы цикла

В Maple имеется несколько операторов цикла, и для их записи используются служебные слова for, from, by, to, while, do, od и end do. Телом всех операторов цикла является последовательность команд, заключенных между do и end do (od – синоним этой комбинации). Начнем с перечислительного цикла, который есть практически во всех алгоритмических языках:

```
for VAR from VAL1 by VAL2 to VAL3 do EXPR end do;
```

Тело цикла EXPR выполняется при каждом значении параметра цикла VAR, который изменяется от VAL1 с шагом VAL2 до тех пор, пока не станет больше VAL3. Отметим, что если шаг изменения VAL2 равен единице, то оператор цикла допускает сокращенную форму:

```
for VAR from VAL1 to VAL3 do EXPR od;
```

А если и начальное значение VAL1переменной VAR равно единице, то возможен еще более короткий вариант:

```
for VAR to VAL3 do EXPR od;
```

Приведем пример оператора цикла:

```
> for i from 0 by 4 to 8 do i; end do;
```

0

4

8

Оператор цикла типа «пока» имеет вид:

```
while BOOL do EXPR od;
```

Тело цикла EXPR выполняется, пока значение логического выражения BOOL истинно (true), и выполнение прекращается, если BOOL ложно (false). Приведем соответствующий пример:

```
> j:=0:
while j<5 do j:=(j+1)^j end do;
```

$j := 1$

$j := 2$

$j := 9$

Следующий оператор цикла является некоторым симбиозом двух предыдущих:

```
for VAR from VAL1 by VAL2
          while BOOL do EXPR end do;
```

Тело цикла EXPR выполняется, пока логическое выражение BOOL является истинным, а переменная VAR изменяется от значения VAL1 с шагом VAL2. Отметим, что Maple вначале проверяет условие цикла, а затем выполняет выражение EXPR. Например:

```
> for x from 1 by 2 while x<6 do x^2 od;
```

1

9

25

Четвертый оператор цикла ориентирован на работу с символьными выражениями и имеет следующую форму:

```
for VAR in EXPR1 do EXPR2 od;
```

Тело цикла EXPR2 (последовательность команд между do и od) выполняется, когда символьная переменная VAR последовательно принимает значения каждого из операндов алгебраического выражения или списка EXPR1. Естественно, работа этой конструкции зависит от внутреннего представления выражения EXPR1. Если EXPR1 — сумма, то переменная VAR принимает поочередно значения каждого слагаемого, если произведение — то каждого сомножителя, и т. д. Поясним сказанное примером:

```
> f:=x^2+3*x+1/x; g:=simplify(f);
```

$$f := x^2 + 3\,x + \frac{1}{x}$$

$$g := \frac{x^3 + 3\,x^2 + 1}{x}$$

```
> for s in f do s; od;
```

x^2

$3x$

$\frac{1}{x}$

```
> for s in g do s; od;
```

$x^3 + 3x^2 + 1$

$\frac{1}{x}$

Команды break и next предназначены для управления операторами цикла. По команде break осуществляется немедленный выход из цикла, а по команде next — переход к следующему шагу. Приведем два примера:

```
> i:=0: do i:=i+1; if i=3 then break fi end do;
```

$i := 1$

$i := 2$

$i := 3$

```
> L:=[1,2,5,100]: for i in L do
                 if i=5 then next end if; i^2 od;
```

1

4

10000

Иногда бывает полезным бесконечный цикл, который можно реализовать, опустив все управляющие циклами параметры:

```
do EXPR end do;
```

Этот цикл будет работать до тех пор, пока внутри него не встретится ошибка или не будет осуществлен выход с помощью команд break или return (см. следующий раздел).

Для обработки возможных ошибок и аварийных или специальных ситуаций существует конструкция вида:

```
try EXPR1
 catch CATCHSTRING:EXPR2
 finally EXPR3
end;
```

Выражение EXPR1 выполняется до тех пор, пока не возникла ошибка или ситуация, описываемая строкой CATCHSTRING. При возникновении такой ситуации выполняется выражение EXPR2, а выражение EXPR3 выполняется в любом случае. В этой конструкции обязательна только первая ветка try. Пример:

```
> i:=2:
> try
  do
  result := j/i;
  print("Try",result);
```

```
    i:=i-1;
    od
catch:
    result := j^2;
    print("Catch",i);
finally i,j^3
end try;
```

$Try, \frac{1}{2}j$

Try, j

$Catch, 0$

$0, j^3$

Подробнее о возможных вариантах строки CATCHSTRING смотрите в справке Maple.

Функции, процедуры и модули

В Maple существует множество команд, которые реализованы в виде процедур-функций, процедур или модулей. В этом разделе мы кратко опишем, как создавать такие конструкции самостоятельно и какие возможности это предоставляет.

Процедуры-функции

Процедуры-функции в Maple можно задавать несколькими способами. Первый напоминает задание отображения — некоторое число входных параметров при помощи знака -> формирует результирующее выражение expr:

```
name:= (var1,var2,...) -> expr;
```

Здесь name — имя процедуры-функции, var1, var2, ... — имена формальных параметров, а expr — выражение, реализующее тело процедуры-функции. Напомним, что типы формальных параметров и результата работы процедуры-функции могут быть любыми.

Приведем пример задания процедуры-функции:

```
> f:=(x,y)->simplify(x^2+y^2);
```

$f := (x, y) \to \text{simplify}(x^2 + y^2)$

```
> f(sin(x),cos(x));
```

1

Второй способ задания процедуры-функции при тех же входных параметрах использует команду unapply:

```
name:=unapply(expr,var1,var2,...);
```

Здесь var1, var2, ... — переменные, а expr — выражение или операция. Эта команда полезна при определении новой функции через известную или когда вычисленное выражение предполагается использовать как процедуру-функцию. Например:

```
> f:=unapply(diff(z(x)^2,x)-2,z);
```

$$f := z \to 2\, z(x) \left(\frac{\partial}{\partial x} z(x) \right) - 2$$

```
> f(sin^2);
```

$$4 \sin(x)^3 \cos(x) - 2$$

С процедурами-функциями в Maple можно проделывать элементарные математические операции, такие как композиция, умножение и др. Приведем примеры:

```
> f:=x->sin(x+2): g:=x->arcsin(x)-2:
> h:=f+g: h(y);
```

$$\sin(y+2) + \arcsin(y) - 2$$

```
> h:=f*g: h(y);
```

$$\sin(y+2)\,(\arcsin(y) - 2)$$

```
> h:=f@g: h(y);
```

$$y$$

Выразительно применение отображения (так называемой функции без имени) в качестве параметра в ряде команд. Приведем пример использования функции при обращении к команде map:

```
> map(x->x^2,[1,2,3,4,5]);
```

$$[1, 4, 9, 16, 25]$$

Процедуры

Всякая процедура в Maple начинается с заголовка, который состоит из имени процедуры, за которым следует знак присваивания и служебное слово proc, затем в круглых скобках через запятую указываются формальные параметры. Процедура должна обязательно заканчиваться оператором end (в последней версии пакета можно использовать синоним end proc). Все команды и выражения, стоящие после заголовка и до оператора end, составляют тело процедуры. Простейшая процедура имеет следующий вид:

```
NAME:=proc( VAR1, VAR2, ...);
EXPR1; EXPR2; ...
end proc;
```

Здесь NAME — имя процедуры, VAR1, VAR2, ... — имена формальных параметров, а EXPR1, EXPR2, ... — выражения, реализующие тело процедуры.

Текст процедуры должен быть набран в одной группе. По нажатию клавиши **Enter** происходит синтаксический анализ текста и в случае ошибки выводится сообщение о ней. После того как процедура загружена в память, к ней можно обращаться по имени. Возвращаемым значением по умолчанию является результат последнего оператора из тела процедуры, однако существуют и другие возможности, о которых будет сказано ниже. Тип результата работы процедуры зависит от типа возвращаемого значения. Например, процедура с именем f, вычисляющая сумму переменных x и y, выглядит следующим образом:

```
> f:=proc(x,y);  x+y;  end;
```

$$f := \textbf{proc}\,(x, y)\; x + y\; \textbf{end proc}$$

Приведем два результата обращения к этой процедуре:

```
> f(u,sin(v)); f(1,9);
```

$u + \sin(v)$

10

Для написания процедур в Maple имеется ряд команд и служебных слов, кроме указанного выше обязательного минимального набора. Эти команды и слова позволяют описывать переменные, управлять выходом из процедуры, сообщать об ошибках.

Формальный параметр процедуры можно явно описать, указав его тип после двух двоеточий, следующих за именем параметра. В этом случае при обращении к процедуре Maple проверит тип фактического параметра и выведет сообщение об ошибке в случае его несовпадения с типом формального параметра. Пример:

```
> g:=proc(i::integer) i^2 end proc:
> g(123456789);
```

15241578750190521

```
> g(11/2);
Error, g expects its 1st argument, i, to be of type integer, but received 11/2
```

Еще одним способом передачи данных из процедуры является изменение значения входных параметров. При помощи описания evaln можно присваивать переменным, выступающим в качестве входных параметров, значения внутри процедуры. Отметим, что присваивать фактическим параметрам возвращаемые значения следует только перед выходом из процедуры, что связано со спецификой обработки выражений внутри процедур. Пример:

```
> f:=proc(x::evaln) x:=10; x; end proc;
```

$f := \mathbf{proc}\,(x{::}evaln)\; x := 10;\, x\;\mathbf{end\;proc}$

```
> f(p); p;
```

p

10

После заголовка процедуры может следовать описательная часть процедуры, отделяющаяся от него пробелом. В этой части описываются локальные и глобальные переменные, используемые процедурой. Локальные переменные доступны только внутри тела процедуры, и они не конфликтуют с одноименными глобальными и локальными переменными других процедур. Следует помнить, что при выполнении операций с локальными переменными производится только один уровень оценивания, см. пример в следующем разделе. Все переменные, описанные во внешнем по отношению к процедуре блоке, являются для нее глобальными. Изменения глобальных переменных внутри тела процедуры действительны и за ее пределами.

Для определения локальных переменных, используемых только внутри данной процедуры, применяется описатель local. Перечислить глобальные переменные можно при помощи описателя global, который должен размещаться в описательной части процедуры. Во избежание накладок с использованием имен рекомендуется описывать все переменные, встречающиеся в процедуре. Если в теле процедуры встречаются неописанные переменные, стоящие в левой части оператора

присваивания или фигурирующие в качестве переменной цикла, то Maple выводит предупреждение и автоматически описывает их как локальные. Приведем пример:

```
> restart; i; j; k; l:=4;
```

i

j

k

$l := 4$

```
> a:=proc() local i; global j;
i:=1; j:=2-1; k:=3;
end proc:
Warning, "k" is implicitly declared local to procedure "a"

> a():
> i; j; k;
```

i

-2

k

После описателей в процедуре может стоять указатель на параметры:

```
options  OPT;
```

Термин OPT может принимать одно из следующих значений: remember (ускоряет выполнение процедур и делает рекурсивные процедуры более эффективными), operator (аналогичен заданию процедуры-функции) и некоторые другие (builtin, system, arrow, angle, trace, package и Copyright). После описателей идет тело процедуры, которое может содержать команды и другие процедуры.

Для выхода из процедуры в любом месте ее тела используется команда RETURN(VAL). Здесь VAL — возвращаемое значение. Если в процедуре несколько операторов RETURN, то возвращаемые значения могут быть даже различных типов.

Для аварийного выхода из процедуры в случае возникновения ошибки и сообщения о случившемся используется команда ERROR("text"), где text — сообщение, которое должно появиться на экране в аварийной ситуации. Для вывода предупреждающих сообщений используется команда WARNING (см/ справку Maple).

Схематично общий вид процедуры можно изобразить следующим образом:

```
NAME:=proc(PAR1::type1,...,PARK::typek)
local VAL1,...,VALN;
global VAR1,...,VARM;
options OPT;
EXPR1; EXPR2;... RETURN(res1); ...
ERROR("Error in procedure name");...EXPRP;
end proc;
```

Здесь EXPR1; EXPR2; ...; EXPRN — группы операторов Maple. Приведем пример, демонстрирующий ряд возможностей при конструировании процедур. Описываемая процедура преобразует отрицательное число в положительное. Если входным па-

раметром является нуль, то выводится сообщение об ошибке; для положительных чисел выводится массив, содержащий само число, его квадрат и куб:

```
> examp:=proc(x::numeric) global z; local y,w;
 options remember;
 if x<0 then RETURN(-x);
 elif x=0 then ERROR("Variable x=0"); fi;
 [x,x^2,x^3];
 end;
```

examp := proc (*x*::*numeric*)
local *y*, *w*;
global *z*;
option *remember*;
if *x* < 0 then RETURN(−*x*) elif *x* = 0 then ERROR(`*Variable x=0*`) end if ;

[*x*, *x*^2, *x*^3]
end proc

Обратимся к процедуре с различными входными параметрами:

```
> examp(-1); examp(0); examp(3);
```

1

```
Error, (in examp) Variable x=0
```

[3, 9, 27]

Продемонстрируем эффективность использования параметров на примере remember, которая сохраняет промежуточные результаты процедуры в памяти. Опишем рекурсивную процедуру fib1, вычисляющую число Фибонначи, и аналогичную ей процедуру fib2, не использующую параметр remember. В процедурах тип nonnegint указывает на целые положительные числа:

```
> fib1 := proc (n::nonnegint) option remember;
   if n < 2 then n else fib1(n-1)+fib1(n-2) end if;
 end proc;
 fib2 := proc (n::nonnegint)
   if n < 2 then n else fib2(n-1)+fib2(n-2) end if;
 end proc;
```

Теперь обратимся к процедурам, выводя время их выполнения:

```
> time(fib1(30)); time(fib2(30));
```

.024

103.033

Эффект использования параметра remember очевиден.

Иногда процедура должна уметь обрабатывать различное число входных параметров. В этом случае при описании процедуры не обязательно указывать явно количество и тип параметров. Число фактических параметров можно узнать внутри тела процедуры при помощи команды nargs, а обращаться к ним нужно как к элементам массива args. Приведем пример процедуры без предварительно описанных параметров:

```
> examp:=proc() local n: n:=nargs:
  if nargs=2 then return "Число параметров равно 2",
           args[1]-args[2] end if:
  [nargs,args] end proc:
```

Обратимся к процедуре с различным числом параметров.

```
> examp(a,b,c): examp(a,b):
```

$[3, a, b, c]$

Число параметров равно $2a - b$

Для вывода дополнительной информации о работе процедуры при ее отладке можно использовать команду userinfo. Первый параметр команды задает номер уровня вложенности, при котором будет выводиться информация, а второй — имя процедуры, в которой находится обращение к команде. Уровень вложенности определяется глобальной переменной infolevel. Поясним сказанное примером:

```
> f:=proc(x,y) userinfo(2,f,"Были введены:",x,y); x^2+y^2 end:
> infolevel[f]:=2:
> f(1,2):
f:   Были введены:   1   2
```

5

Обработка процедур и возможные ошибки

При работе в Maple могут возникать ошибки, которые связаны со спецификой обработки обращений к процедурам. При обращении к процедуре Maple в первую очередь анализирует на правильность ее имя, затем входные параметры, после чего выполняет тело процедуры. Число фактических параметров может быть меньше числа описанных параметров процедуры. В этом случае ошибка возникнет только тогда, когда недостающий параметр понадобится для выполнения команд. Например:

```
> f:=proc(a,b)
   if a<0 then print("Значение первого параметра
                    отрицательно");
          else print(cat("Значение второго параметра: ",
                        convert(b,string)));
   end if;
end proc:
> f(-1):
```

Значение первого параметра отрицательно

```
> f(1):
Error, (in f) f uses a 2nd argument, b, which is missing
> f(1,3):
```

Значение второго параметра: 3

При обращении к процедурам следует помнить, что многие операции выполняются только с переменными определенного типа и важно соответствие типа фактического параметра типу, требуемому в операторах тела процедуры. Например, использование в качестве фактического параметра процедуры f из предыдущего примера символьной переменной a приведет к ошибке при проверке условия:

```
> f(a,b):
Error, (in f) cannot evaluate boolean: a < 0
```

Если бы переменной a перед обращением к f было присвоено числовое значение, то ошибка бы не возникла. Ошибки можно избежать, используя обращение к процедуре как к процедуре отложенного действия, указав ее имя в кавычках. При таком обращении не происходит выполнения тела процедуры. Результат можно присвоить переменной и вернуться к выполнению процедуры после явного определения всех фактических параметров.

Например:

```
> s:='f'(a,b);
```

$s := \mathrm{f}(a)$

```
> a:=1: s;
```

Значение второго параметра: b

Еще одним источником ошибок может быть использование в процедурах локальных переменных. Дело в том, что для оценки локальных переменных Maple использует только один уровень оценивания и это иногда приводит к неверным результатам. Проиллюстрируем сказанное примером:

```
> f:=proc(x)
   local a,b,c;
   a:=b;
   b:=c;
   a+x;
end proc;
```

$f := \mathbf{proc}\,(x)\ \mathbf{local}\ \ a, b, c;\ a := b;\ b := c;\ a + x\ \mathbf{end\ proc}$

```
> f(1);
```

$b + 1$

Этой ошибки можно избежать, если перед оператором a+x; поставить оператор a:=eval(a).

Модули

Если процедуры предназначены для создания команд, включающих в себя набор Maple-команд, то модули позволяют создавать более сложные конструкции, такие как наборы процедур, данных и др. Описание модуля имеет вид:

```
module ()
export ESEQ;
local LSEQ; global GSEQ;
option OPTSEQ; description DESC;
BODY
end module .
```

Описатель export предназначен для задания имен процедур, наборов данных и других составляющих модуля, которые будут доступны для вызова. Обращение к этим объектам осуществляется при помощи команды выбора (:-). Другие описатели аналогичны соответствующим описателям процедур.

Приведем пример простых процедуры и модуля, которые реализуют операции сложения и произведения двух величин:

```
> expr:=proc()
     local plus,prod;
        plus:=(a,b)->a+b;
        prod:=(a,b)->a*b;
        plus,prod
     end proc:
```

Обратимся к процедуре expr и выведем результат:

```
> s1:=expr(); s1[1](1,1); s1[2](1,1);
```

s1 := *plus*, *prod*

2

1

Теперь приготовим модуль для выполнения аналогичных операций:

```
> exmo:=module()
     export plus,prod;
        plus:=(a,b)->a+b;
        prod:=(a,b)->a*b;
     end module:
Обратимся к элементам модуля:
> exmo:-plus(1,1); exmo:-prod(1,1);
```

2

1

Модули являются мощным инструментом программирования в Maple и позволяют реализовывать самые разнообразные конструкции. Последнее время широкое распространение получили объектно-ориентированные языки программирования. Хотя Maple и не является таким языком в полной мере, но с помощью модулей можно реализовывать разнообразные операции с объектами. Приведем пример конструктора объекта «комплексное число»:

```
> ComNum:=proc(rel,img)
   if nargs<>2 then
        error("Неправильный входной параметр") fi;
   module()
     export re,im,abs,arg;
     local real,imag;
     real,imag:=rel,img;
     re:=()->real;
     im:=()->imag;
     abs:=()->sqrt(re()^2+im()^2);
     arg:=()->arctan(im(),re());
   end module;
 end proc:
```

Создадим с помощью конструктора ComNum комплексное число 2+2I и выведем его модуль и мнимую часть.

```
> z:=ComNum(2,2);
```

$z := \mathbf{module}\ ()\ \mathbf{local}\ real, imag; \mathbf{export}\ re, im, \text{abs}, arg;\ \mathbf{end\ module}$

```
> z:-abs(); z:-im();
```

$2\sqrt{2}$

2

Макроопределения

Часто бывает удобно ввести новые имена для уже существующих функций или переменных. Это позволяет сделать команда alias. Проиллюстрируем ее действие примером:

```
> alias(F=sin):
> F(x); simplify(subs(x=Pi/2,%));
```

$\mathrm{F}(x)$

1

Для задания макроопределений (макросов) используется команда macro. Ее удобно применять для введения сокращенных имен-синонимов команд Maple, присвоения одного имени для последовательности команд или переопределения уже существующих команд. Поясним сказанное примерами. Сначала определим макрос с именем cc для вычисления определителя при помощи команды det пакета linalg и обратимся к нему:

```
> restart; macro(cc=linalg[det]):
> cc([[a1,a2],[a3,a4]]);
```

$a1\ a4 - a2\ a3$

Следующий макрос реализует сразу несколько операций:

```
> macro(cs=abs@(cos+"-"+sin)):
> g:=cs(x);
```

$g := |-\cos(x) + x - \sin(x)|$

Подставим в полученное выражение значение x:

```
> subs(x=0.1,g);
```

$|-\cos(.1) + .1 - \sin(.1)|$

Значение выражения вычислено не было, для этого требуется выполнить команду evalf. Переопределим команду подстановки subs так, чтобы после подстановки всегда выполнялись действия с плавающей запятой:

```
> macro(subs=evalf@subs);
```

c, cs, subs

Теперь при обращении к команде subs сразу производится и вычисление выражения:

```
> subs(x=0.1,g); subs(x=1/5,(x-1)^3);
```

$.9948375820$

$-.5120000000$

Создание и использование пакетов и библиотек

Пакетом обычно является коллекция процедур, предназначенных для решения определенного класса задач. В поставке Maple существует более двадцати пакетов, в том числе пакет для решения задач линейной алгебры, пакет по теории графов и др. Этот стандартный набор можно расширять пакетами, разработанными самим пользователем. В создаваемый пакет имеет смысл помещать уже отлаженные и оттестированные процедуры. Напомним, что один из вариантов обращения к команде name_com пакета **name_pack**, выглядит следующим образом: name_pack[name_com](options). Таким образом, полное имя команды, входящей в пакет, состоит из имени этого пакета и имени самой команды, а пакет представляет собой как бы массив, элементами которого являются команды.

Для оформления пакета нужно переменной с составным именем присвоить процедуру. После того как пакет написан, его нужно сохранить на диске в файле с расширением .m (например, с именем namefile) при помощи команды

```
save name_pack,"namefile.m"
```

В последующих сеансах пакет нужно предварительно считать с диска при помощи команды

```
read "namefile.m"
```

Пакеты можно создавать также при помощи модулей (фактически модуль и является пакетом). Для подключения пакета используется команда with.

Приведем пример небольшого пакета **pack1** из трех процедур с именами proc1, proc2, proc3:

```
> pack1[proc1]:=
 proc(a) local s; s:=2*a; print("proc1: ",s); s; end proc:
 pack1[proc2]:=
 proc(a) local s; s:=a^2; print("proc2: ",s); s; end proc:
 pack1[proc3]:=
 proc(a,b) local s; s:=(a+b)/2; print("proc3: ",s); s:
        end proc:
```

Сохраним пакет в файле **pack1.m**:

```
> save pack1,"pack1.m";
```

Теперь в другом сеансе мы можем считать этот пакет командой read и подключить при помощи команды with:

```
> restart:
> read "pack1.m":
> with(pack1);
```

$[proc1, proc2, proc3]$

```
> proc2(a):
```

proc2: , a^2

Если пользователем написаны и используются несколько собственных пакетов, то целесообразно создать собственную библиотеку. В предыдущих версиях Maple для

этого была предназначена программа march.exe. В состав ядра Maple 6 включена команда march. С ее помощью можно создавать библиотеки, добавлять или исключать элементы библиотек и т. д. Обращение к ней имеет вид:

```
march(command,archive_dir,options)
```

Здесь command описывает действие с библиотекой archive_dir, а options — дополнительные параметры. Мы не будем далее подробно останавливаться на описании работы с библиотеками, а ограничимся только простым примером. Для более подробной информации следует обратиться к справке Maple.

Вначале создадим новую библиотеку (директория c:\temp\ должна уже существовать на диске) при помощи команды

```
> march("create","c:\\temp\\",1);
```

Отметим, что при указании директорий в Maple вместо одного используется два обратных слеша.

Для работы с библиотеками в Maple имеются две зарезервированные переменные: libname — содержит пути к каталогам, где находятся доступные в текущем сеансе библиотеки, а savelibname — имя каталога с библиотекой, в которую будут записываться новые процедуры. Файлы с библиотеками имеют имя maple.lib. Для того чтобы в сеансе стала доступна дополнительная библиотека, необходимо этим переменным присвоить строки с указанием пути к библиотеке:

```
> libname:=libname,"c:\\temp\\";
```

libname := "C:\\PROGRAM FILES\\MAPLE 6/lib"," "c:\\temp\\"

```
> savelibname:=libname[2]:
```

Приготовим модуль, содержащий две процедуры:

```
> exmo:=module()
      export plus,prod;
      local a,b;
      option package;
          plus:=proc(a,b) a+b end proc;
          prod:=proc(a,b) a*b end proc;
      end module:
```

Сохраним его и пакет pack1 (будем считать, что мы его уже подключили) в библиотеку с помощью команды savelib:

```
> savelib(exmo,pack1);
```

Чтобы записанные команды активировать в другом сеансе, нужно добавить путь к пользовательской библиотеке в список libname и подключить пакеты при помощи команды with. Например:

```
> restart;
> libname:=libname,"c:\\temp\\":
> with(exmo);
```

[*plus*, *prod*]

В Maple имеется возможность сопровождать написанные команды текстами справки. Для этого используется команда makehelp:

```
makehelp(topic, txtfile, library)
```

Здесь topic — имя команды, для которой создается справка, txtfile — имя текстового файла, содержащего текст справки, library — имя библиотеки, содержащей команду.

Команды ввода/вывода

В Maple можно сохранить в файле результаты всего сеанса работы, но часто бывает нужно записать только конечный результат, а не весь ход решения. Это становится особенно актуальным, если для получения результата проводился большой объем символьных вычислений и манипуляций с формулами. Для записи данных в Maple существует команда save. Если результатами работы являются значения переменных var1, var2, ..., то для сохранения их в файл с именем name и расширением ext достаточно дать команду

```
save var1,var2,...,"name.ext";
```

Способ представления сохраняемой информации в файле зависит от расширения ext. Если в качестве расширения указано .m, то файл запишется во внутреннем Maple-формате, при других расширениях команды Maple запишутся в текстовом формате. Например, в результате выполнения команд

```
> a:=sin(x): save a,"a.m";
```

будет создан файл с именем a.m со следующей информацией:

```
M6R0
I"a-%$sinG6#%"xG6"
```

Приведем пример сохранения в текстовом формате:

```
> a:=sin(x): save a,"a.txt";
```

Содержимое файла имеет вид:

```
a := sin(x);
```

Если использовать команду save без перечисления переменных, а указать только имя файла, то произойдет сохранение всех назначенных за сеанс величин.

Для ввода информации из файла с именем name и расширением ext применяется команда

```
read "name.ext";
```

Для считывания строки из файла можно использовать команду

```
readline ("name.ext");
```

Эту команду можно применять и для организации интерактивного ввода, указав в качестве имени файла terminal. После этого Maple будет ожидать ввода строки с клавиатуры. Например:

```
> a:=readline(terminal);
> Это пример интерактивного ввода
```

a := **"Это пример интерактивного ввода"**

Понятно, что после набора вводимой строки следует нажать клавишу Enter.

Для записи результатов работы в файл имеются команды

```
writeto ("name.ext")
```

и

```
appendto ("name.ext") .
```

После выполнения этих команд все вводимое и результаты работы команд будут записаны в файл с именем name.ext. Тип файла по-прежнему зависит от указываемого расширения ext. По команде writeto информация записывается с начала нового файла, а использование appendto означает запись в конец уже существующего файла. Для восстановления вывода на экран нужно повторно обратиться к команде writeto, указав в качестве имени файла terminal (то есть после команды writeto("terminal") все результаты снова выводятся на экран).

Существует также возможность записи результатов в файл при помощи следующих команд:

- writebytes (file,expr1) — записать выражение expr1 в файл с именем file в байтовом виде;
- writeline ("name.ext",str) — запись строки str в файл с именем name.ext.

Естественно, перед тем как файл использовать, его нужно открыть, а по завершении работы закрыть при помощи команд:

- open ("name.ext",mode) — открыть файл name.ext. В качестве параметра mode могут выступать WRITE (открыть на запись) и READ (открыть на чтение):
- close ("name.ext") — закрыть файл name.ext.

Часто при написании программы необходимо выводить информацию о ходе решения, результаты, аварийные сообщения и пр. Для этих целей в пакете предусмотрен ряд команд печати. Наиболее простой является команда print, обращение к которой имеет вид:

```
print(expr1,expr2,...,exprn)
```

Здесь expr1,expr2,...,exprn — любые Maple-выражения. Если переменной ничего не присвоено, то печатается просто имя переменной, в противном случае печатается ее содержимое. Приведем пример обращения к команде печати:

```
> x:=y^2: print(x,"Information",y,factor(x-3*y)):
```

y^2, *Information*, y, $y\,(y-3)$

В отличие от команды print, которая печатает выражения через запятую в естественном математическом виде, команда lprint выводит информацию в стиле строки ввода и разные выражения отделяются друг от друга пробелами. Например:

```
> x:=y^2: lprint(x,"Information",y,factor(x-3*y));
y^2, Information, y, y*(y-3)
```

Помимо бесформатного вывода в Maple есть команда printf(fmt,expr1,expr2,...). Здесь fmt — спецификация формата вывода, и другие команды печати, полностью идентичные аналогичным командам языка С и MATLAB. Более подробно о них можно прочитать в главе 16 «Программирование в MATLAB».

Пакет Maple можно использовать для анализа и графической интерпретации числовой информации, находящейся в текстовом файле и полученной при помощи как самого пакета, так и других программ. Обычно в текстовом файле числа запи-

саны в строки, по несколько чисел в строке. Для считывания числовой информации из файла используется команда

```
readdata (name,options,posint)
```

Здесь name — имя файла, options — тип переменных (integer/float), posint — счетчик чисел (сколько считывать чисел из строки). Прочитанная информация представляется в виде переменной типа listlist.

Например, пусть в текстовом файле a.txt в первой строке находятся числа: 0, 1, 2, 3, 4, а во второй — 5, 6, 7, 8, 9. Считаем данную информацию из файла и занесем ее в переменную data:

```
> data:=readdata("a.txt",integer,5);
```

$data := [[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]]$

Двойная индексация у переменной data связана с тем, что числа записываются в переменную типа listlist, так что первая размерность переменной равна числу считанных строк, а вторая определяется параметром posint.

Табулированную информацию в текстовый файл можно записать командой

```
writedata (fileID, data, format, default)
```

Здесь fileID является именем файла, переменная data содержит записываемую информацию, необязательный параметр format указывает формат записи (integer/float/string), а в качестве default можно указывать процедуру форматирования вывода (см. справку ?writedata). Приведем пример записи гильбертовой матрицы третьего порядка в файл:

```
> writedata("hilb.txt",linalg[hilbert](3),float):
```

В результате в файле hilb.txt будет содержаться следующая информация:

```
1               .5            .3333333333
.5              .3333333333   .25
.3333333333     .25           .2
```

Для считывания строки из файла name.ext применяется команда readline("name.ext"), а для записи — writeline(file,str). Результат действия первой команды присваивается переменной типа string. Если в качестве имени файла указать terminal, то программа будет ожидать ввода с клавиатуры.

Кроме того, в пакете stats имеются команды readstat и writestat для ввода и вывода статистической информации, а пакет LinearAlgebra включает две команды для импорта и экспорта матриц:

```
ImportMatrix (f, s, fo, d, de, tr)
ExportMatrix (f, M, t, de, tr)
```

Здесь f — строка символов с именем файла, M — имя переменной, содержащей матрицу, s — параметр, имеющий вид source=name и определяющий формат считывания из файла, параметр t имеет вид target=name и определяет формат записи в файл. В качестве термина name у последних двух параметров могут выступать Matlab, MatrixMarket или delimited. Параметр fo имеет вид format=name (здесь name одно из имен rectangular/entries/vectors) и управляет методом сохранения информации, d имеет вид datatype=type и определяет тип данных, с помощью параметра de (delimiter=text) можно задать разделитель между числами, а параметр tr (true/

false) указывает на необходимость транспонирования матрицы. Обязательными являются только параметры f, M, s.

В заключение коснемся возможностей записи рисунков в файл. При работе в Windows-версии это легко сделать, отметив рисунок, затем нажав на правую кнопку мыши и выбрав в появившемся меню пункт export. Другую возможность предоставляет команда interface с параметрами plotdevice=val. Здесь строковой переменной val можно указать тип вывода графики (gif — в формате gif, laserjet — в формате HP LaserJet принтера и т. д.). Задать имя файла для вывода графики можно при помощи конструкции: plotoutput="namefile.ext". Например, после выполнения команд

```
> interface(plotdevice=gif,plotoutput="ggg.gif");
  plot3d(sin(x*y)*y,x=-4..4,y=-4..4);
```

на диске появится файл ggg.gif с графической копией рисунка в формате gif. Обратим внимание на то, что, пока значение plotoutput не переопределено, результаты всех команд графики будут записываться в один файл, замещая предыдущий следующим. Следует отметить, что копия рисунка в файле не всегда будет идентична варианту на экране, что, по-видимому, объясняется спецификой реализации.

Вместо команды interface можно использовать:

```
plotsetup (DeviceType, TerminalType, options...)
```

Здесь DeviceType указывает графический формат (gif, ps, bmp), а необязательный параметр TerminalType определяет тип устройства (имя файла). Информацию о наборе параметров смотрите в справке пакета. Например, для того, чтобы рисунки сохранялись в файле с именем a.bmp с портретной ориентацией страницы, нужно выполнить следующую команду:

```
> plotsetup(bmp,plotoutput="a.bmp",plotoptions="portrait");
```

Возможности работы с файлами не ограничиваются перечисленными командами, информацию о других командах можно найти в справке пакета.

Отладка программ

В этом пункте мы рассмотрим возможности отладчика программ и коснемся получения более подробной информации о стандартных Maple-командах, ходе вычислений, используемых ресурсах компьютера и ошибках, а также перечислим команды управления выводом результатов.

Информация о переменных и объектах

Чтобы выяснить, присвоено ли переменной с именем name какое-либо значение, существует команда assigned(name). Результатом работы этой команды является истина (true) или ложь (false). Например:

```
> restart; a:=10; assigned(a); assigned(sin); assigned(b);
```

true

true

false

Команда anames выводит назначенные в текущем сеансе имена. Если вместо параметров команды стоят пустые скобки, то выводятся определенные в сеансе переменные, если внутри скобок указан тип переменных — все назначенные имена, имеющие значения этого типа. Приведем пример:

```
> restart:
> a:=10: b:=a/c:
anames(); anames("*");
```

a, b

b

```
> anames(integer);
```

Digits, a, printlevel, Order

В Maple существует целый ряд глобальных системных констант и команд, управляющих работой программы. О некоторых из них уже шла речь в предыдущих главах, о некоторых будет сказано ниже. Список системных констант и команд можно вывести командой:

```
> anames("environment");
```

Testzero, UseHardwareFloats, Rounding, %, %%%, Digits, index/newtable, mod, %%, Order, printlevel, Normalizer, NumericEventHandlers

Вообще говоря, каждая назначенная переменная Maple является объектом определенного типа (о типах см. первые главы книги), который описывается свойствами. Свойства можно просмотреть, используя команду op, обсуждавшуюся во второй главе книги, которая в Maple также является объектом. Для работы со структурой сложных объектов (таблиц, массивов, процедур и т. д.) к ним предварительно нужно применять команду оценивания eval. Проиллюстрируем просмотр свойств процедуры и в качестве примера рассмотрим команду sin. Сначала выясним тип объекта и количество его свойств:

```
> restart:
> whattype(eval(sin));
```

procedure

```
> nops(eval(sin));
```

7

Всякий объект типа procedure имеет семь свойств, выведем первые четыре из них для нашего примера:

```
> op(1,eval(sin));
```

x::algebraic

```
> op(2,eval(sin));
```

n, t, pull_out, keep_in

```
> op(3,eval(sin));
```

Copyright (c) 1992 by the University of Waterloo. All rights reserved.

```
> op(4,eval(sin));
```

$$\text{table}([0 = 0, \infty = \mathit{undefined}, \frac{1}{4}\pi = \frac{1}{2}\sqrt{2}, -\infty = \mathit{undefined}, \pi = 0, I = I\sinh(1), \frac{1}{6}\pi = \frac{1}{2},$$

$$\frac{1}{2}\pi = 1,$$

$$\frac{1}{3}\pi = \frac{1}{2}\sqrt{3}$$

])

Итак, первым по счету свойством процедуры является список параметров, вторым — список локальных переменных, третьим — параметры процедуры и четвертым параметром является таблица запоминаемых значений (remember table). Пятое, шестое и седьмое свойства содержат соответственно строку описания, список глобальных переменных и лексическую таблицу.

Информация о работе команд и обработка ошибок

Начнем с системной переменной printlevel, которая задает детальность печати сообщений о ходе преобразований. Многие процедуры Maple вызывают внутри себя другие команды, которые называются вложенными. Если значение переменной printlevel находится в интервале от 1 до 5, то выводится информация о данной команде, если от 6 до 10 — то дополнительно выводится информация о вызываемой вложенной команде первого уровня, и т. д. Для того чтобы выводилась информация о работе команды с уровнем вложенности k, значение printlevel должно быть равно 5*k. По умолчанию printlevel=1, а это значит, что выводится результат только непосредственно вызываемой команды. При значении printlevel=1000 выводится информация о работе всех вложенных команд.

Для получения подробной информации о работе некоторой команды comm можно использовать вызов trace(comm) или debug(comm); после этого при обращении к команде comm будет выводиться информация о ее работе. Например:

```
> trace(diff):
> printlevel:=6:
> diff(x*sin(x),x);
execute diff, args = x*sin(x), x
{--> enter sin, args = x
<-- exit sin (now at top level) = sin(x)}
{--> enter diff/sin, args = x, x
execute diff, args = x, x
```

$\cos(x)$

```
<-- exit diff/sin (now at top level) = cos(x)}
```

$\sin(x) + x\cos(x)$

Отменить действие команды trace можно при помощи команды untrace, а debug — соответственно undebug.

Управлять представлением результата работы команд на экране можно при помощи команды interface(ARG1,ARG2,...), где параметры ARG имеют вид NAME=VAL. Приведем некоторые возможные значения переменных NAME и VAL. Беря в качестве NAME имя prettyprint и задавая число VAL, пользователь может управлять видом стандартных сообщений на экране, например, при prettyprint=0 используется команда lprint. Указание verboseproc=n задает степень подробности печати текстов процедур: при n=0 текст не печатается, при n=1 печатается только текст пользовательских процедур, при n=2 печатаются все процедуры.

Иногда необходимо посмотреть тексты стандартных Maple-команд. Чтобы вывести на экран текст процедуры с именем comm, нужно определить параметры вывода командой interface и дать команду

```
print(comm)
```

Приведем набор команд, в результате которых выводится текст команды int:

```
> interface(verboseproc=2,prettyprint=1,version):
> print(int);
  proc()
local answer;
option "Copyright (c) 1997 Waterloo Maple Inc. All rights reserved.";
    if nargs = 2 and
    type(args[2], "="(name, {complex(float) .. algebraic, algebraic .. complex(float)}))
    then try return evalf("int"(args)) catch:  end try
    end if;
    if evalb(_EnvInt96) = true and evalb(_EnvInt98) = true then
        error "The flags _EnvInt96 and _EnvInt98 cannot be both set to true"
    elif evalb(_EnvInt96) = true then "int96/int"(args)
    elif evalb(_EnvInt98) = true then "int98/int"(args)
    else
        answer := "int/int"([args], Digits, _EnvCauchyPrincipalValue);
        if answer = FAIL then error "wrong number (or type) of arguments" end if;
        answer
    end if
end proc
```

Для обработки ошибок вычислений в Maple предусмотрены команда traperror и системная переменная lasterror. В переменной lasterror сохраняется сообщение о последней случившейся ошибке, причем содержимое переменной очищается при обращении к traperror. Поясним сказанное примером:

```
> 1/0;
Error, division by zero
> lasterror; 1; lasterror;
```

"division by zero"

1

"division by zero"

```
> traperror(1);lasterror;
```

1

lasterror

Для обработки событий, происходящих при действиях с числами с плавающей запятой, существует специальная системная команда NumericEventHandler, которая позволяет обрабатывать нестандартные ситуации. Обратимся к этой команде без параметров и получим список таких ситуаций вместе с установками по умолчанию:

```
> NumericEventHandler();
```

invalid_operation = default, division_by_zero = default, overflow = default,
underflow = default, inexact = default, real_to_complex = default

Эти установки можно переопределить, описав собственную процедуру обработки ситуации. Приведем пример процедуры, которая при возникновении деления на ноль выводит предупреждение, а в качестве результата выводит 10^1000:

```
> div_zero := proc(opr,opd)
WARNING("Деление на ноль в %1 с параметрами %2", opr,opd);
   10.0^1000 end proc:
NumericEventHandler(division_by_zero=div_zero);
```

division_by_zero = default

```
> (sin(x)+cos(y))/0.0;
Warning, Деление на ноль в / с параметрами [1, 0.]
```

$$.1000000000\ 10^{1001} \sin(x) + .1000000000\ 10^{1001} \cos(y)$$

Ряд команд позволяет следить за использованием ресурсов компьютера. Так, команда `time()` показывает время процессора в секундах, используемое с начала текущего сеанса работы с Maple. Если в скобках в качестве параметра указана команда, то будет выведено время выполнения указанной команды. Приведем пример:

```
> restart;
> time();
```

13.015

```
> time(int(sum(sin(x^i),i=1..10),x));
```

20.564

Для ограничения времени выполнения конкретной команды можно использовать команду `timelimit`. Первым параметром этой команды является максимальное время выполнения выражения, а вторым — само выполняемое выражение. При превышении заданного времени Maple прервет выполнение выражения и выведет сообщение об ошибке. Проиллюстрируем работу этой команды примером. Опишем процедуру, которая не предусматривает выхода из нее, и обратимся к ней с использованием ограничения времени:

```
> restart;
> f:=proc(i::evaln) i:=0;
  while true do i:=eval(i+1) od;
  end:
> timelimit(0.05,f(j));
Error, (in f) time expired
```

Теперь выведем количество выполненных внутри процедуры циклов:

```
> j;
```

9137

Время выполнения каждой команды будет показываться после запуска команды `showtime()`. После выполнения этой команды Maple переходит в режим, при котором выводится информация о затраченных каждой командой ресурсах. При этом изменяется приглашение к вводу — вместо символа > появляются символы 01, 02 и т. д. Для отмены действия режима достаточно ввести команду `off`.

Для контроля над установками и ресурсами ядра Maple предназначена команда `kernelopts`, которая в зависимости от указанных параметров выводит различную

информацию о текущих ресурсах. В качестве параметров могут выступать: ASSERT, assertlevel, bytesalloc, bytesused, cputime, dagtag, dirsep, gcbytesavail, gcbytesreturned, gcfreq, gctimes, inline, level, maxdigits, maximmediate, memusage, platform, printbytes, printlevel, profile, stacklimit, student, unread, version, wordsize. За подробной информацией о каждом параметре следует обращаться к справке Maple. Приведем пример, в котором с помощью описанной команды выведем информацию о версии пакета, используемой платформе и максимально возможной мантиссе числа:

```
> kernelopts(version,platform,maxdigits);
```

Maple 6, IBM INTEL NT, Jan 31 2000 Build ID 16401, "windows", 268435448

Команда kernelopts с параметром profile вместе с командой exprofile позволяют включить режим профилирования в Maple. После включения этого режима возможен просмотр разнообразной информации о сеансе. Приведем пример:

```
> restart;
> a:=proc(x) sin(x)+x end:
b:=proc(x) cos(x)-a(x) end:
> kernelopts(profile=true):
  writeto("output");
  b(1);
  kernelopts(profile=false);
  writeto(terminal);
  exprofile("output",alpha);
9 different functions, using 36.180 secs and 11994K words
name                    #calls        cpu                words
====               ======      ===               =====
Main_Routine             1  36.163 (100.0%)  11 986303 ( 99.9%)
a                        1    .007 (    .0%)        190 (    .0%)
b                        1   0.000 (   0.0%)        295 (    .0%)
cos                      1    .003 (    .0%)        548 (    .0%)
csgn                     2   0.000 (   0.0%)        336 (    .0%)
readlib                  3    .007 (    .0%)        364 (    .0%)
readstore                3   0.000 (   0.0%)       5842 (    .0%)
sin                      1   0.000 (   0.0%)        203 (    .0%)
type/SymbolicInfinity 1  0.000 (   0.0%)        235 (    .0%)
```

Для профилирования конкретной функции служит команда profile. Для сбора информации об интересующей команде достаточно обратиться к profile, указав в качестве параметра имя команды. Для вывода собранной информации используется команда showprofile, а для отмены режима профилирования — unprofile. В качестве примера зададим режим профилирования для двух описанных выше процедур a и b, обратимся к ним и выведем собранную информацию:

```
> profile(a,b): b(2):
> showprofile(a);
function      depth     calls     time      time%    bytes    bytes%
---------------------------------------------
a               1          1      0.000      0.00     8604     30.52
---------------------------------------------
total:          2          2      .004     100.00    28188    100.00
> unprofile(a,b);
```

Работа с отладчиком программ

В состав предыдущих версий пакета входила программа mint.exe, которая позволяла диагностировать написанные программы, анализировать синтаксические ошибки, использование переменных и оптимальность работы процедур, а также задавать точки отладки внутри процедур. В Maple 6 эти возможности входят в состав ядра пакета. В режиме отладки становится доступным целый ряд команд, часть из которых мы перечислим, а полный их список можно найти в справке по пакету.

Для вывода информации об отладке имеются команды:

- showstat(COMM) — выводит на экран текст процедуры COMM с номерами команд;
- showstop — выводит информацию о точках отладки во всех процедурах, введенных в текущем сеансе.

Для начала отладки команды COMM необходимо выполнить команду

```
stopat(COMM)
```

Тогда при обращении к команде COMM пакет переходит в режим, в котором возможно пошаговое выполнение операторов процедуры, просмотр содержимого ее локальных и глобальных переменных и т. д. Переход к режиму отладки в сеансе обозначается приглашением к вводу DBG>. Режим отладки отменяется командой

```
unstopat(COMM)
```

Если необходимо перейти в режим отладки только при обращении к определенному месту процедуры, то перед нужным оператором в тексте необходимо поставить команду

```
DEBUG("text")
```

Необязательный параметр "text" содержит текстовую строку с сообщением. Если отладка нужна для контроля глобальной переменной VAR, то используется команда

```
stopwhen(VAR)
```

При этом пакет перейдет в отладочный режим в том месте, где встречается переменная VAR. В ходе отладки можно использовать следующие команды (полный список см. в справке пакета):

- next — выполнение следующей команды текущего уровня;
- step — выполнение следующего шага любого уровня;
- sont — продолжение выполнения программы до следующей точки отладки;
- where — вывод информации о текущем уровне отладки;
- quit — выход из отладчика.

Приведем пример использования отладчика. Определим процедуру с именем f, задав в ней одну точку отладки (оператор DEBUG):

```
> f := proc(x,y) local a; global b;
    if x < y then
       DEBUG("ветвь 1"); a := x; b := y + a;
    else
       a := y; b := x + a;
    end if;
```

```
    a + b + x + y
  end proc:
```

Используем некоторые из описанных команд для анализа работы этой процедуры. Сначала посмотрим структуру введенной процедуры:

```
> showstat(f);
f := proc(x, y)
local a;
global b;
   1   if x < y then
   2     DEBUG("ветвь 1");
   3     a := x;
   4     b := y+a
       else
   5     a := y;
   6     b := x+a
       end if;
   7   a+b+x+y
end proc
```

Теперь посмотрим информацию обо всех точках отладки в данном сеансе:

```
> showstop();
Breakpoints in:
   f
No variables being watched.
No errors being watched.
```

Вызовем процедуру f, что приведет к переходу в режим отладки при обращении к команде DEBUG("ветвь 1"):

```
> f(1,2);
"ветвь 1"
f:
   3     a := x;
```

Последние две строки вывода указывают место, где остановлен процесс вычислений. Посмотрим в отладчике содержимое переменной y:

```
DBG> y;
2
f:
   3     a := x;
```

Продолжим вычисления:

```
DBG> cont;
 7
```

Теперь определим режим отладки для всей процедуры f:

```
> stopat(f);
```

После этого при обращении к f пакет переходит в режим отладки:

```
> f(2,1);
f:
```

```
  1*  if x < y then
        ...
      else
        ...
      end if;
```

Выполним один шаг процедуры:

```
DBG> step;
f:
   5     a := y;
```

А теперь выйдем из отладчика:

```
DBG> quit;
Warning, computation interrupted
```



Математические библиотеки Maple

О возможностях пакетов аналитических (символьных) вычислений судят по их умению решать самые разнообразные математические задачи. Кроме команд стандартной библиотеки Maple оснащен большим количеством команд, организованных в пакеты, специализированные по темам.

В предыдущих главах были рассмотрены команды и пакеты, без которых, по нашему мнению, невозможно содержательное введение в пакет Maple и квалифицированная работа с ним. Подробное изложение в рамках данной книги всех команд стандартной библиотеки и наполнения других пакетов не представляется возможным, но информация о них необходима. Поэтому в этой главе мы опишем некоторые команды стандартной библиотеки, не рассмотренные ранее, и дадим эскизные характеристики большинства пакетов. Напомним, что для обращения к командам пакета с именем package его нужно загрузить при помощи команды

```
with(package)
```

В табл. 8.1 дан перечень математических пакетов Maple с указанием глав настоящей книги, в которых имеется описание или приведены обзорные сведения об этих пакетах.

Таблица 8.1. Список пакетов Maple

Имя пакета	Назначение пакета
algcurves	Алгебраические кривые, см. раздел «Алгебры и формы» этой главы
codegen	Оптимизация текстов программ на языке Maple и перевод Maple-выражений в конструкции языков программирования, см. главу 9 «Maple и другие программы»
combinat	Комбинаторика, см. эту главу
combstruct	Комбинаторные структуры, см. эту главу
context	Создание контекстных меню для работы в диалоговом режиме, см. справку Maple
DEtools	Исследование обыкновенных дифференциальных уравнений, см. главу 4 «Решение уравнений в Maple»
diffalg	Команды дифференциальной алгебры, см. раздел «Алгебры и формы» этой главы

Имя пакета	Назначение пакета
difforms	Дифференциальные формы, см. раздел «Алгебры и формы» этой главы
Domains	Конструирование и анализ сложных алгоритмов, см. эту главу и справку Maple
finance	Финансовая математика. Короткое описание дано в этой главе
GaussInt	Работа с Гауссовыми целыми числами, см. раздел «Теория чисел» данной главы
genfunc	Работа с рациональными производящими функциями, см. справку Maple
geom3d	Команды трехмерной евклидовой геометрии, см. раздел «Стереометрия» данной главы
geometry	Команды двумерной евклидовой геометрии, см. раздел «Геометрия на плоскости» данной главы
GF	Работа с конечными полями Галуа, см. эту главу
Groebner	Работа с полиномами с использованием базиса Гребнера, см. эту главу
group	Группы перестановок, см. раздел «Комбинаторика» этой главы
inttrans	Интегральные преобразования, см. главу 3 «Математический анализ в Maple»
liesymm	Группы и симметрии Ли, см. данную главу
linalg	Линейная алгебра. Большая часть команд пакета описана в главе 5 «Алгебра в Maple»
LinearAlgebra	Пакет для решения задач линейной алгебры, основанный на новых структурах данных Maple (rtable и module) и использующий алгоритмы NAG. Большая часть команд пакета описана в главе 5 «Алгебра в Maple»
LREtools	Работа с линейными рекуррентными уравнениями, см. главу 4 «Решение уравнений в Maple»
Matlab	Пакет, обеспечивающий связь систем Maple и MATLAB, см. главу 9 «Maple и другие программы»
networks	Операции теории графов, см. данную главу
numapprox	Приближение функций, см. главу 3 «Математический анализ в Maple»
numtheory	Теория чисел, см. эту главу
Ore_algebra	Вычисления в алгебрах линейных операторов, см. эту главу
orthopoly	Ортогональные полиномы, см. данную главу
padic	Работа с р-адическими числами, см. раздел «Теория чисел» данной главы
PDEtools	Исследование уравнений в частных производных, см. главу 4 «Решение уравнений в Maple»
plots	Двумерная и трехмерная графика, см. главу 6 «Графика Maple»
plottools	Работа с графическими объектами, см. главу 6 «Графика Maple»
polytools	Операции с полиномами, см. главу 2 «Аналитические преобразования в Maple»
powseries	Работа с формальными степенными разложениями, см. главу 3 «Математический анализ в Maple»
process	Функции для создания многозадачных программ Maple. Информацию о нем см. в справке Maple
simplex	Линейная оптимизация, см. данную главу
slode	Решение линейных обыкновенных дифференциальных уравнений в виде формальных рядов, см. справку Maple
Spread	Команды для работы с электронными таблицами Maple, см. соответствующий раздел данной главы

продолжение ⇗

Таблица 8.1 (*продолжение*)

Имя пакета	Назначение пакета
stats	Математическая статистика, см. эту главу
student	Пакет учебных вычислений. Многие команды описаны в главе 3 «Математический анализ в Maple»
sumtools	Операции с конечными и бесконечными суммами, см. справку Maple
tensor	Тензорные операции и решение задач общей теории относительности, см. соответствующий раздел данной главы

Пакет финансовой математики

В этом пакете собраны команды для финансовых расчетов, к которым относятся вычисление финансовых потоков, сложного процента и ряд других. Не описывая этих команд, ограничимся примерами использования некоторых из них (понятных авторам, не экономистам по специальности). Примеры сформулируем в виде небольших задач. Вначале подключим сам пакет и получим список всех его команд:

```
> with(finance);
```

[*amortization*, *annuity*, *blackscholes*, *cashflows*, *effectiverate*, *futurevalue*, *growingannuity*, *growingperpetuity*, *levelcoupon*, *perpetuity*, *presentvalue*, *yieldtomaturity*]

Задача 1. На счете имеется 800 рублей. Сколько рублей будет на счете через 5 лет при ставке 7 % годовых? Ответ:

```
> futurevalue(800,0.07,5);
```

1122.041385

Задача 2. Когда вы станете миллионером, если при ставке 5 % годовых положили в банк 1000 рублей? Ответ:

```
> solve(futurevalue(1000,0.05,t)=1.0e6,t);
```

141.5808985

Задача 3. Сколько нужно положить денег в банк сегодня, чтобы при ставке 15 % годовых иметь на счете через 10 лет 1 000 000 рублей? Ответ:

```
> presentvalue(1000000,0.15,10);
```

247184.7061

Задача 4. Какой должен быть годовой доход в процентах, чтобы за десять лет стать миллионером, имея 100 рублей? Ответ:

```
> u:=fsolve(presentvalue(1e6,x,10)=100,x)*100;
```

$u := 151.1886432$

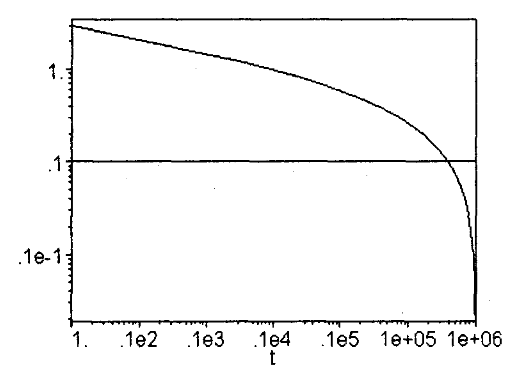
Проверим вычисления при помощи команды `futurvalue`:

```
> futurevalue(100,u*0.01,10);
```

$.1000000002\ 10^7$

Теперь построим в логарифмическом масштабе график, показывающий, как должен зависеть от величины начального капитала годовой банковский процент, обеспечивающий один миллион рублей через десять лет. Прямой линии на графике соответствует «разумный» доход в 10 % годовых:

```
> plots[loglogplot]({0.1,
"fsolve(presentvalue(1e6,x,10)=t,x)"},t=1..1.0e6,axes=boxed);
```



Задача 5. Каковы потери за год работника бюджетной сферы с месячной зарплатой 1000 рублей при инфляции 2 % в месяц? Ответ:

```
> cashflows([seq(1000,i=1..12)],0.02)-12000;
```

-1424.65878

Геометрические пакеты

В Maple есть два геометрических пакета: geometry — для задач планиметрии и geom3d — для задач стереометрии. Перед обращением к их командам сами пакеты должны быть загружены при помощи команды with. В каждом пакете кроме команд задания геометрических объектов (линии, плоскости, окружности, сферы и т. д.) имеются команды для определения некоторых характерных величин (площади, объема и др.), а также ряда менее стандартных величин (например, линий Эйлера). Все геометрические структуры, определенные при помощи одного из этих пакетов, могут использоваться только в пределах действия данного пакета. Нельзя, например, в команде пакета geom3d ссылаться на точку, заданную при помощи команды из пакета geometry.

Для всех геометрических пакетов характерен следующий способ определения объектов: первый параметр команды задает имя объекта и далее следует собственно информация об объекте. Этим геометрические пакеты отличаются от обычного в Maple определения какой-нибудь переменной или структуры при помощи операции присваивания. Для просмотра полей структуры, описывающей геометрический объект, нужно использовать команду detail.

Например, подключим пакет geometry и определим точку с координатами (1,2):

```
> with(geometry):
> point(p,[1,2]);
```

p

Теперь при помощи команды detail узнаем имя объекта, тип и другие его характеристики:

```
> detail(p);
```

name of the object: p
form of the object: point2d
coordinates of the point: [1, 2]

В обоих пакетах для графического вывода геометрического объекта используется команда draw. При выводе на одном рисунке нескольких геометрических объектов можно указывать параметры графического вывода (например, цвет) каждого объекта. Результатом команды являются структуры двумерной или трехмерной графики, и поэтому при обращении к команде draw используются параметры, аналогичные графическим (см. главу 6 «Графика Maple»). Далее коротко охарактеризуем команды геометрических пакетов.

Геометрия на плоскости

Пакет geometry содержит команды для решения задач двумерной евклидовой геометрии. Перед началом работы пакет нужно загрузить. По умолчанию _x и _y используются как глобальные переменные для координат точек, а также в качестве переменных в уравнениях прямых и окружностей. Геометрические объекты определяются обычным образом: точка задается своими координатами (команда point), прямая — двумя точками или уравнением (команда line), окружность (circle) — тремя точками, уравнением, заданием центра и радиуса, диаметром. Команды определения объектов пакета geometry даны в табл. 8.3.

Ряд команд пакета проверяет выполнение того или иного условия для одного или нескольких геометрических объектов. Мнемоника здесь вполне понятная, и при возможности определенного ответа результатом является булевская константа (true или false); в некоторых случаях выводятся координаты объекта (например, точки), при которых будет выполнено проверяемое условие. Приведем в табл. 8.2 список команд, опуская параметры, если их количество и назначение следуют из пояснения.

Таблица 8.2. Команды проверки условий для двумерных геометрических объектов

Имя	Условие
AreCollinear	Лежат ли три точки на одной прямой
AreConcurrent	Проходят ли три прямые через одну точку
AreConcyclic	Проверка существования окружности, которой принадлежат заданные четыре точки
AreHarmonic	Проверка двух точек на гармоническую сопряженность двум другим точкам
AreOrthogonal	Ортогональны ли два геометрических объекта
AreParallel	Параллельны ли две прямые
ArePerpendicular	Перпендикулярны ли две прямые

Имя	Условие
AreSimilar	Подобны ли два треугольника
AreTangent (line,circle)	Является ли прямая line касательной окружности circle
IsEquilateral	Проверка треугольника на равносторонность
IsRightTriangle	Является ли треугольник прямоугольным
IsOnCircle(pt,circle)	Принадлежит ли точка pt окружности circle
IsOnLine(pt,line)	Принадлежит ли точка pt прямой line

При задании большинства геометрических объектов (вершины треугольника, точки отрезка и др.) никакие символьные переменные использоваться не должны. Отметим, что обращение ко многим командам может иметь несколько вариантов. Например, окружность может задаваться набором точек, координатой центра и радиусом или уравнением. Для многих команд результат действия присваивается переменной с именем name. Приведем только представительное подмножество таких команд, см. табл. 8.3, а полный их список можно найти в справке Maple.

Таблица 8.3. Команды определения двумерных геометрических объектов и действий с ними

Имя	Назначение
area	Вычисление площади заданного объекта (треугольника, круга или квадрата)
bisector(bs,pt,tri)	Вычисление отрезка bs от вершины pt до середины противоположной стороны треугольника tri. Синонимом этой команды является median
center(name,circle)	Определение центра окружности, результат присваивается переменной name
centroid(name, tri)	Вычисление центра тяжести треугольника
circle(name, [pt,expr], [namex,namey])	Вычисление окружности с центром в точке pt и радиусом expr. По умолчанию центру окружности присваивается имя center_c. [namex,namey] определяют имена переменных по осям
circle(name, [pt1,pt2,pt3])	Вычисление окружности, проходящей через три точки. Существует также возможность определять окружность уравнением
circumcircle (name,tri)	Вычисление описанной вокруг треугольника tri окружности
convexhull	Вычисление окружности, проходящей через три точки из заданного множества точек так, что все остальные точки содержатся внутри окружности
coordinates(pt)	Вывод координат точки pt
detail(arg)	Вывод информации об аргументе arg, в качестве которого может быть геометрический объект
diagonal(Sq)	Вычисление длины диагонали квадрата Sq
diameter([pt1,pt2,…]))	Вычисление диаметра круга, содержащего заданные точки
distance(pt,line)	Вычисление расстояния между точкой pt и прямой line. В качестве второго параметра может фигурировать точка, тогда вычисляется расстояние между двумя точками

продолжение ⇒

Таблица 8.3 (*продолжение*)

Имя	Назначение
draw(obj)	Рисование графического объекта
ellipse	Определение эллипса одним из следующих способов: по пяти точкам, по центру и двум полуосям или при помощи уравнения
foci(focn,name)	Вычисление фокусов эллипса или гиперболы
FindAngle	Вычисление угла между двумя прямыми или двумя окружностями
Hyperbola	Определение гиперболы, задаваемой набором точек или другими характеристиками
incircle(name, tri)	Вычисление вписанной в треугольник tri окружности
intersection (pt,obj1,obj2)	Вычисление точки пересечения двух прямых или двух окружностей
inversion (name,obj,circle)	Вычисление для объекта obj инверсии относительно окружности circle
line	Определение прямой, заданной двумя точками или уравнением
median(name,A,tri)	Определение медианы треугольника tri, проведенной из вершины A
midpoint(name,pt1,pt2)	Вычисление средней точки на отрезке, заданном двумя точками pt1 и pt2
parabola(name,...)	Задание параболы набором точек или другими характеристиками
ParallelLine (name,pt,line)	Вычисление прямой, проходящей через точку pt и параллельной прямой line
PerpenBisector (name,pt1, pt2)	Вычисление прямой, проходящей через середину отрезка, заданного двумя точками pt1 и pt2, и ортогональной ему
PerpendicularLine (name,pt, line)	Вычисление прямой, проходящей через точку pt и перпендикулярной прямой line
point(name,a,b)	Задание точки с координатами a и b
projection (name,pt,line)	Вычисление проекции точки pt на прямую line
radius(circle)	Вычисление радиуса окружности
randpoint(name,line)	Задание случайной точки на прямой line
reflect(name,obj,a)	Вычисление объекта, зеркально симметричного объекту obj относительно прямой или точки a
rotation(name,obj, angl,dir,c)	Вычисление результата вращения геометрического объекта obj на угол angl в направлении dir относительно центра вращения c
sides	Вычисление периметра треугольника или квадрата
square(name, [pt1,pt2,pt3,pt4])	Задание квадрата четырьмя точками
TangentLine (name,pt,circle)	Вычисление двух прямых, проходящих через точку pt и касательных к окружности circle; результат присваивается переменной name
Tangentpc (name,pt,circle)	Вычисление касательной к окружности circle, проходящей через точку pt
Translation (name,obj,AB)	Перенос объекта obj направленным отрезком AB
triangle	Задание треугольника тремя точками, тремя прямыми или тремя сторонами

Поясним действие ряда команд примером. Подключим пакет geometry и различными способами зададим окружности и определим их центры:

```
> with(geometry):
> circle(c,(x-3)^2 + (y-1)^2 =4,[x,y]):
  circle(c1,[point(A,0,0), point(B,2,0),
          point(C,1,2)],'centername'=O1):
> center(c), coordinates(center(c));
```

center_c, [3, 1]

```
> center(c1), coordinates(center(c1));
```

$O1, \left[1, \frac{3}{4}\right]$

Теперь введем три точки A, B, F и определим проходящие через них прямые AB, AF. Используем разделитель «двоеточие» для отмены вывода на экран результата работы некоторых команд:

```
> point(A,[0,1]): point(B,[1,0]): point(F,[1,1]):
```

A

```
> line(AB,[A,B]): line(AF,[A,F]):
```

AB

Прямую CD введем с помощью уравнения

```
> line(CD,x+y=2,[x,y]):
```

Проверим параллельность введенных прямых:

```
> AreParallel(AB,CD); AreParallel(AB,AF);
```

true

false

Выясним, лежит ли точка F на прямой CD:

```
> IsOnLine(F,CD);
```

true

Вычислим расстояние от точек A и F до прямой CD:

```
> distance(F,CD); distance(A,CD);
```

0

$\frac{1}{2}\sqrt{2}$

Теперь определим треугольник ABS и найдем его площадь:

```
> triangle(ABS,[A,B,point(S,[-1,-2])]):
> area(ABS);
```

2

Проведем окружность, проходящую через середины сторон треугольника ABS:

```
> EulerCircle(Elc, ABS, "centername"=cn):
```

Elc

Выведем характеристики созданного геометрического объекта:

```
> detail(Elc);
assume that the names of the horizontal and vertical axes are _x and _y, respectively
```

name of the object: Elc
form of the object: circle2d
name of the center: cn
coordinates of the center: [1/4, -1/4]
*radius of the circle: 1/8*5^(1/2)*8^(1/2)*

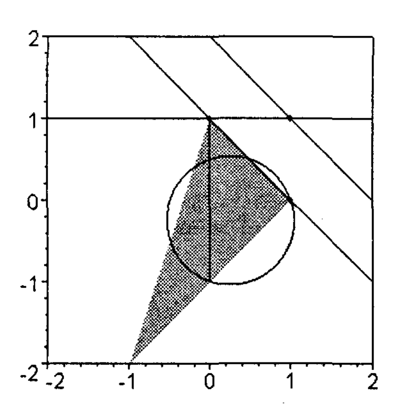
*equation of the circle: _x^2-1/2+_y^2-1/2*_x+1/2*_y = 0*

Определим медиану треугольника ABS, проведенную из вершины A:

```
> median(mA, A, ABS,M);
```

В заключение выведем все созданные в этом примере объекты на одном графике:

```
> draw({A,B,F,AB,CD,AF,ABS(filled=true,color=green),mA,Elc},
     view=[-2..2,-2..2],color=black);
```



Стереометрия и пакет geom3d

Команды пакета трехмерной геометрии geom3d похожи на рассмотренные команды двумерной геометрии. Мы не будем приводить повторно команды, которые отличаются добавлением только одной координаты в параметрах. Как и в предыдущем разделе, ограничимся представительным подмножеством команд пакета, а полный их список можно найти в справке Maple.

Для определения точки, прямой, плоскости и сферы применяются соответственно функции point, line, plane и sphere. Можно также определить отрезок (segment), направленный отрезок (dsegment), треугольник (triangle) и целый ряд многогранников (например, командой tetrahedron создается пирамида). Как и в предыдущем разделе, при задании геометрических объектов возможны различные способы их определения (координатами точек, уравнением и др.). Для вывода информации об объекте используется команда detail, а для его изображения — команда draw. В пакете geom3d по умолчанию идентификаторы _x, _y, _z и _t используются для указания координат точек, а также в уравнениях прямых, плоскостей и сфер. Приве-

дем пример задания сферы, пирамиды и плоскости, которая проходит через центр сферы:

```
> with(geom3d):
> sphere(sphr,[point(osphr,0,0,0),2]);
```

sphr

```
> tetrahedron(tet,point(otet,0,0,5),3):
> plane(pln,[osphr,[0,0,-0.1]]):
```

После этого выведем информацию о сфере и нарисуем все три геометрических объекта:

```
> detail(sphr);
Warning, assume that the name of the axes are _x, _y and _z
```

name of the object: sphr

form of the object: sphere3d

name of the center: osphr

coordinates of the center: [0, 0, 0]

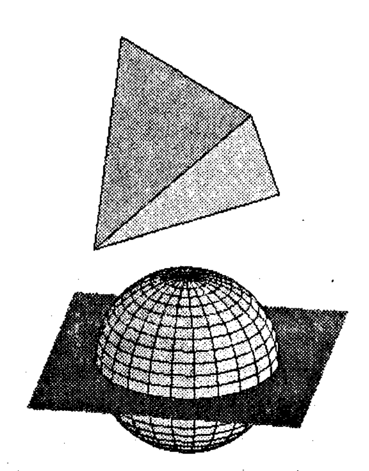
radius of the sphere: 2

*surface area of the sphere: 16*Pi*

*volume of the sphere: 32/3*Pi*

equation of the sphere: _x^2+_y^2+_z^2-4 = 0

```
> draw({sphr,tet,pln(color=blue,style=PATCHNOGRID)},
     orientation=[10,65]);
```



Ряд команд выполняет проверку некоторых геометрических условий, см. табл. 8.4. В следующих командах, см. табл. 8.5, если это не оговорено особо, результат действия присваивается первому параметру команды (переменной с именем name). Задание большинства геометрических объектов (вершин треугольника или тетраэдра, точек отрезка и др.) не должно содержать никаких символьных переменных.

Таблица 8.4. Команды проверки условий для объектов трехмерной геометрии

Имя	Условие
AreCollinear	Лежат ли три точки на одной прямой
AreConcurrent	Проходят ли три прямые через одну точку
AreCoplanar	Принадлежат ли четыре точки или две прямые одной плоскости
AreParallel	Проверка параллельности двух прямых, двух отрезков или двух плоскостей
ArePerpendicular	Проверка перпендикулярности двух прямых, двух отрезков или двух плоскостей
IsOnObject(pnt.obj)	Принадлежит ли точка pnt геометрическому объекту obj
IsTangent(pl.sph)	Является ли плоскость pl касательной к сфере sph

Таблица 8.5. Команды определения трехмерных геометрических объектов и действий с ними

Имя	Назначение
area	Вычисление площади треугольника, сферы или многогранника
center	Вычисление центра геометрического объекта
coordinates	Вывод координат точки
distance	Вычисление расстояния между двумя геометрическими объектами. В качестве таких объектов могут фигурировать точки, прямые, отрезки или плоскости
Equation	Вывод уравнения геометрического объекта
FindAngle	Вычисление угла между двумя геометрическими объектами
form	Вывод информации о виде геометрического объекта
inter	Вычисление пересечения двух или трех геометрических объектов
midpoint	Вычисление середины отрезка, соединяющего две точки
parallel	Вычисление плоскости или прямой, проходящей через точку и параллельной заданной плоскости или прямой
projection	Вычисление проекции одного геометрического объекта на другой
radius	Вычисление радиуса объекта
reflection	Вычисление геометрического объекта, зеркально симметричного данному относительно точки, прямой или плоскости
sides	Вывод величины ребра правильного многогранника
TangentPlane (name.pt.sph)	Вычисление плоскости name, касательной к сфере sph и проходящей через точку pt
tetrahedron	Определение тетраэдра по четырем точкам
translation	Перенос геометрического объекта, заданный направленным отрезком
vertices	Вывод координат вершин многогранника
volume	Вычисление объема сферы или многогранника

Проиллюстрируем действие ряда команд и начнем с команд преобразования графических объектов. Сначала определим три точки A, B и C, прямую l1 и направленный отрезок. L:

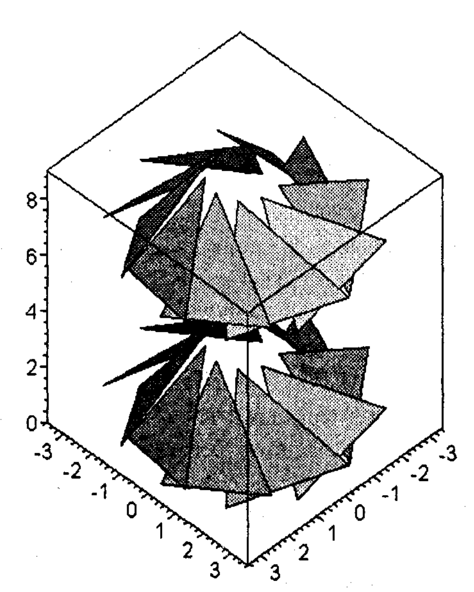
```
> point(A,3,0,0),point(B,2,3,1),point(C,1,0,3):
 line(li,[0,0,t],t):
 dsegment(L,point(l1,0,0,0),point(l2,0,0,6)):
```

Зададим треугольник T1 по трем точкам A, B и C. После этого вычислим множество треугольников, полученных вращением на различные углы треугольника T1 вокруг прямой li. Третья команда формирует множество треугольников, полученных переносом треугольников первого набора на отрезок L:

```
> triangle(T1,[A,B,C]):
  seq(rotation(T1r||i,T1,evalf(Pi/5*i),li),i=1..10):
  seq(translation(T1t||i,T1r||i,L),i=1..10):
```

Теперь нарисуем все полученные треугольники:

```
> draw({seq(T1r||i,i=1..10),seq(T1t||i,i=1..10),T1},
                                            axes=boxed);
```



При определении некоторых геометрических объектов допускается использование символьных параметров. Для работы с такими объектами, могут понадобиться дополнительные предположения о свойствах символьных параметров. Зададим сферу S, вычислим ее радиус, площадь и объем, а затем проверим, принадлежит ли ей точка A. Результатом последней команды является условие принадлежности точки сфере. Вычисленные радиус и объем будут зависеть от символьной константы r2. Для определения сферы и корректной работы команд необходимо сообщить о положительности параметра r2:

```
> assume(r2>0);
> sphere(S,x^2+y^2+z^2-r2=0,[x,y,z]);
```

S

```
> radius(S); area(S); volume(S);
```

$\sqrt{r2\sim}$

$4\pi r2\sim$

$\frac{4}{3}\pi r2\sim^{(3/2)}$

```
> IsOnObject(point(A,[-1,1,sqrt(r2-2)]),S);
```

true

Интерполяция и аппроксимация

В главе 3 «Математический анализ в Maple» уже рассматривалась аппроксимация функций, а для числовых данных в Maple имеется несколько команд, реализующих обычную и сплайн-интерполяцию, а также метод наименьших квадратов для приближения данных. В результате выполнения любой из этих команд формируется выражение, которое затем можно преобразовать в процедуру. Напомним, что для оформления выражения в виде процедуры можно использовать команду unapply или команду из пакета codegen

```
fproc:=codegen[makeproc](f,x).
```

Здесь fproc — имя формируемой процедуры, x — независимая переменная.

Для построения интерполяционного многочлена относительно переменной var по таблице, заданной векторами X,Y, используется команда interp(X,Y,var). Массивы, задающие узлы интерполяции, могут быть не упорядочены, но массив X не должен содержать одинаковых элементов. Приведем пример:

```
> pnts:=[0,1,2,3,4],[0,-5,9,3,-2];
```

$pnts := [0, 1, 2, 3, 4], [0, -5, 9, 3, -2]$

```
> g:=interp(pnts,x);
```

$$g := \frac{5}{2}x^4 - \frac{43}{2}x^3 + \frac{113}{2}x^2 - \frac{85}{2}x$$

Построение сплайна с переменной var по таблице, заданной векторами X, Y, производится при помощи команды spline(X,Y,var,d). Здесь параметр d определяет порядок сплайна, который может быть линейным (linear), квадратичным (quadratic), кубическим (cubic) и четвертой степени (quartic). По умолчанию строится кубический сплайн. Результатом действия команды будет построение сплайна в виде кусочно-гладкой функции (piecewise). Построим кубический сплайн по определенному выше набору точек pnts:

```
> f:=spline(pnts,x,cubic);
```

$$f := \begin{cases} -\frac{323}{28}x + \frac{183}{28}x^3 & x < 1 \\ \frac{283}{14} - \frac{2021}{28}x + \frac{849}{14}x^2 - \frac{383}{28}x^3 & x < 2 \\ -\frac{2277}{14} + \frac{5659}{28}x - \frac{153}{2}x^2 + \frac{257}{28}x^3 & x < 3 \\ \frac{981}{7} - \frac{2819}{28}x + \frac{171}{7}x^2 - \frac{57}{28}x^3 & \textit{otherwise} \end{cases}$$

Преобразуем полученное выражение в процедуру:

```
> fproc:=codegen[makeproc](f,x);
```

fproc := **proc** (x)
piecewise (x < 1, −323/28×x + 183/28×x^3, x < 2,
283/14 − 2021/28×x + 849/14×x^2 − 383/28×x^3, x < 3,
−2277/14 + 5659/28×x − 153/2×x^2 + 257/28×x^3,
981/7 − 2819/28×x + 171/7×x^2 − 57/28×x^3)

end proc

Другим способом приближения данных является метод наименьших квадратов. Результатом применения этого метода является функция заданного вида, наименее уклоняющаяся от исходных точек. Для применения этого метода в Maple можно воспользоваться командами линейной алгебры или использовать команду `leastsquare` из пакета **stats**. Входными параметрами команды `leastsquare` являются имена переменных, вид функции и набор точек. Построим методом наименьших квадратов приближение кубическим полиномом тестового набора данных `pnts`. Предварительно подключим пакет статистики:

```
> with(stats):
> h:=fit[leastsquare[[x,y], y=a1*x^3+a2*x^2+a3*x+a4]]([pnts]);
```

$$h := y = -\frac{3}{2}x^3 + \frac{53}{7}x^2 - \frac{95}{14}x - \frac{6}{7}$$

Выделим правую часть полученного выражения:

```
> h:=rhs(h):
```

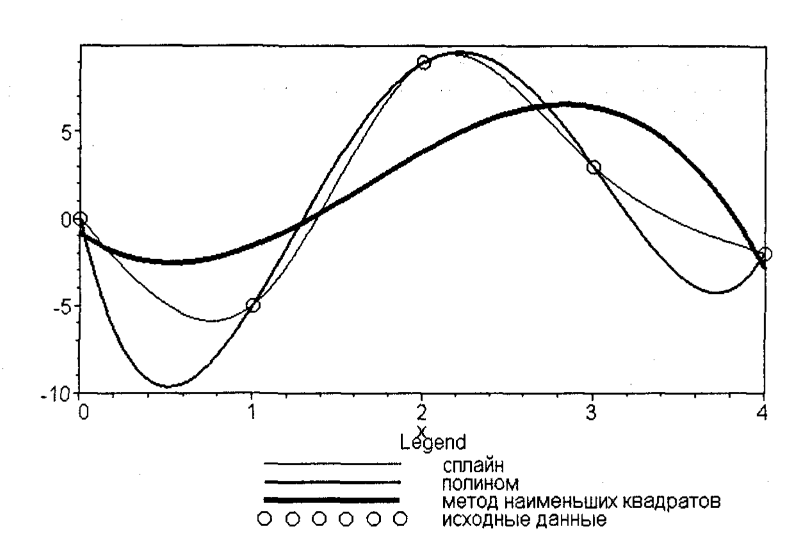
Сравним три полученные различные аппроксимации. Сначала вычислим значения приближений в точке $x=3.5$, которая лежит между узлами интерполяции:

```
> fproc(3.5);
 value(subs(x=3.5,g));
 value(subs(x=3.5,h));
```

-.26339290
-3.2812500
3.830357143

Теперь построим в одних осях графики полученных приближений на рассматриваемом интервале и аппроксимируемые данные:

```
> pic1:=plot([f,g,h],x=0..4,thickness=[0,2,4],color=black,
   legend=["сплайн","полином","метод наименьших квадратов"],
                                                    axes=boxed):
> pic2:=plot([seq([pnts[1][i],pnts[2][i]],i=1..5)],
  style=point,symbol=circle,symbolsize=20,
  color=black,legend=["исходные данные"]):
  plots[display](pic1,pic2);
```



Ортогональные полиномы

Ортогональные полиномы, играющие важную роль в математике, представлены в Maple в виде набора команд стандартной библиотеки, а также собранием процедур пакета orthopoly. Ниже, при перечислении команд, первыми в табл. 8.6 даны команды стандартной библиотеки, а вторыми — соответствующие команды пакета orthopoly. В таблице использованы следующие обозначения: целое число n определяет порядковый номер полинома, a — параметр, x — переменную.

Таблица 8.6. Команды задания ортогональных полиномов

Имя команды стандартной библиотеки	Имя команды пакета orthopoly	Назначение
GegenbauerC(n,a,x)	G(n,a,x)	Полином Гегенбауэра (семейство ультрасферических полиномов)
HermiteH(n,x)	H(n,x)	Многочлен Эрмита
LaguerreL(n,x)	L(n,x)	Многочлен Лагерра
LaguerreL (n,a,x)	L(n,a,x)	Обобщенный многочлен Лагерра
	P(n,x)	Многочлен Лежандра
JacobiP(n,a,b,x)	P(n,a,b,x)	Многочлен Якоби
ChebyshevT(n, x)	T(n,x)	Обобщенный многочлен Чебышева
ChebyshevU(n,x)	U(n,x)	Многочлен Чебышева второго рода

Более детальное описание каждого полинома из пакета orthopoly может быть получено при помощи команды справки ?orthopoly[letter], где letter означает начальную букву соответствующего полинома.

Рассмотрим простой пример. Подключим пакет, выведем полиномы Лежандра второго и пятого порядка, а затем убедимся в их ортогональности на интервале [−1,1] с весовой функцией ($1-x^2$):

```
> restart: with(orthopoly):
```

$[G, H, L, P, T, U]$

```
> P(2,x):P(5,x):
```

$\frac{3}{2}x^2 - \frac{1}{2}$

$\frac{63}{8}x^5 - \frac{35}{4}x^3 + \frac{15}{8}x$

```
> int(P(2,x)*P(5,x)*sqrt(1-x^2),x=-1..1);
```

0

Двумя разными способами определим полином Гегенбауэра второго порядка и вычислим его значение в точке x=1:

```
> g1:=G(2,1,x); g2:=GegenbauerC(2.0,1.0,x);
```

$g1 := 4\,x^2 - 1$

$g2 := \text{GegenbauerC}(2.0,\ 1.0,\ x)$

```
> subs(x=1,g1); value(subs(x=1.0,g2));
```

3

3.000000000

Команды работы с электронными таблицами

При описании интерфейса Maple уже упоминались электронные таблицы Maple и кратко описывалась работа с ними в интерактивном режиме. С этими таблицами можно работать и с помощью команд пакета Spread. Продемонстрируем на примере работу представительного множества его команд. Напомним, что адрес (ссылка на ячейку таблицы) ячейки таблицы состоит из буквы, определяющей столбец таблицы, и цифры, задающей номер строки. Подключим пакет Spread и получим список всех его команд:

```
> with(Spread);
```

[*CopySelection*, *CreateSpreadsheet*, *EvaluateCurrentSelection*, *EvaluateSpreadsheet*, *GetCellFormula*, *GetCellValue*, *GetFormulaeMatrix*, *GetMaxCols*, *GetMaxRows*, *GetSelection*, *GetValuesMatrix*, *InsertMatrixIntoSelection*, *IsStale*, *SetCellFormula*, *SetMatrix*, *SetSelection*]

Создадим электронную таблицу с именем Tbl:

```
> CreateSpreadsheet(Tbl):
```

В результате появится таблица с пустыми ячейками, для краткости мы не приводим соответствующий рисунок.

	A	B	C	D	E
1	0	1	2	$\frac{x^{(\sim B1+1)}}{\sim B1+1}$	
2	1	2	3	$\frac{x^{(\sim B2+1)}}{\sim B2+1}$	
3	2	3	4	$\frac{x^{(\sim B3+1)}}{\sim B3+1}$	
4	$\sim A1^2 + \sim A2^2 + \sim A3^2$				
5					

Рис. 8.1. Заполненная таблица

	A	B	C	D	E
1	0	1	2	$\frac{1}{2}x^2$	
2	1	2	3	$\frac{1}{3}x^3$	
3	2	3	4	$\frac{1}{4}x^4$	
4	5				
5					

Рис. 8.2. Результат выполнения таблицы

Определим матрицу M и присвоим значения ее элементов ячейкам таблицы:

```
> M:=Matrix([[0,1,2],[1,2,3],[2,3,4]]):
> SetMatrix(Tbl,M):
```

В ячейке A4 определим формулу для вычисления суммы квадратов первых трех ячеек столбца A и определим формулы для первых трех ячеек столбца D:

```
> SetCellFormula(Tbl,A4,sum("~A||i"^2,i=1..3)):
> seq(SetCellFormula(Tbl,D||i,int(x^~B||i,x)),i=1..3):
```

После выполнения перечисленных команд таблица примет вид, изображенный на рис. 8.1. Серая штриховка ячеек информирует о том, что заданные для них действия еще не были выполнены.

Выделим элементы столбца A и дадим команду выполнить операции, заданные в выделенных ячейках. Результат выполнения опустим:

```
> SetSelection(Tbl, 1, 1, 4, 1):
> EvaluateCurrentSelection(Tbl):
```

Теперь выполним действия во всей таблице. Результат дан на рис. 8.2:

```
> EvaluateSpreadsheet(Tbl):
```

В завершение отметим, что имеются возможности взаимодействия Maple с электронными таблицами Microsoft Excel. Краткое описание таких возможностей дано в следующей главе.

Пакет тензорного исчисления tensor

Пакет tensor предназначен для проведения тензорных операций и решения задач общей теории относительности (ОТО). Для представления данных имеется тип tensor_type, позволяющий работать с ковариантными и контравариантными компонентами тензоров. Фактически это таблица с двумя полями — для запоминания компонент и описания типа величины. Поле компонент дается массивом, размерность которого отвечает рангу тензора, а индексы изменяются в диапазоне от 1 до размерности рассматриваемого пространства. Для представления типа компонент используется список, в котором положительная единица обозначает контравариантную компоненту, а отрицательная — ковариантную. Для проверки объекта на принадлежность тензорному типу используется команда tensor_type, результатом выполнения которой будет булевская константа (true или false).

Тензору отвечают таблица коэффициентов вращений и таблица компонент кривизны. Для вычисления вращений применяется команда npspin. Для индексации используются греческие буквы. Компоненты кривизны вычисляются по команде npcurve и помещаются в таблицу с тремя полями: компоненты Риччи обозначаются буквой Ф (Phi) и содержатся в массиве размерности (0..2)r(0..2); компоненты Вейля Ψ (Psi) находятся в массиве (0..4); скаляр Риччи обозначен буквой R.

Для упрощения тензорных выражений имеется команда simp. Приведем перечень команд с краткими пояснениями в табл. 8.7.

Таблица 8.7. Команды пакета tensor

Имя	Назначение
Christoffel1	Задание символа Кристоффеля первого рода
Christoffel2	Задание символа Кристоффеля второго рода
Einstein	Задание тензора Эйнштейна
display_allGR	Вывод и описание ненулевых компонент всех тензоров ОТО
displayGR	Вывод ненулевых компонент тензора ОТО
tensorGR	Вычисление тензора кривизны
Jacobian	Якобиан преобразования координат
Killing_eqns	Вычисление компонент (коэффициентов) для уравнения Киллинга
Levi_Civita	Вычисление псевдотензора Леви–Чивита
Lie_diff	Вычисление производной Ли от тензора по ковариантному векторному полю
Ricci	Задание тензора Риччи
Ricciscalar	Задание скаляра Риччи
Riemann	Задание тензора Римана
RiemannF	Задание тензора кривизны Римана
Weyl	Задание тензора Вейля

Далее рассмотрим ряд операций с тензорами, причем перечислено только их представительное подмножество, а полный список нужно смотреть в справке Maple.

Таблица 8.8. Команды, реализующие операции с тензорами

Имя	Назначение
act	Применение операции к элементам тензора
antisymmetrize	Получение антисимметричного тензора по заданным индексам
change_basis	Преобразование базиса тензора
compare	Сравнение двух тензорных объектов
contract	Свертка тензора по одной или нескольким парам индексов
create	Создание тензорного объекта
Jacobian	Вычисление якобиана и обратного к нему преобразования
partial_diff	Вычисление частной производной тензора по заданной координате
prod	Внутреннее и внешнее тензорное произведение
symmetrize	Симметризация тензора по заданным индексам
transform	Преобразование тензора, заданное соответствующим якобианом

Рассмотрим короткий пример:

```
> with(tensor):
> T:=create([1,-1], array([[w,x,0],[y,z,0],[0,y^2,x*y*w]]));
```

$$T := \mathrm{table}([compts = \begin{bmatrix} w & x & 0 \\ y & z & 0 \\ 0 & y^2 & x\,y\,w \end{bmatrix}, index_char = [1, -1]])$$

```
> U:=contract(T,[1,2]):
```

$U := \text{table}([compts = w + z + x\,y\,w,\ index_char = [\ ]])$

```
> partial_diff(U,[x,y,z]);
```

$\text{table}([compts = [y\,w,\ x\,w,\ 1],\ index_char = [-1]])$

```
> act(subs,[x=1],%);
```

$\text{table}([compts = [y\,w,\ w,\ 1],\ index_char = [-1]])$

Теория чисел

Для эффективной работы с задачами теории чисел в Maple имеется ряд команд стандартной библиотеки, а также пакеты numtheory, GaussInt и padic. Входными параметрами и результатами действия команд этих пакетов выступают числа — целые, рациональные, цепные дроби, гауссовы целые (комплексные целые), p-адические, простые, а также несколько известных иррациональных чисел.

Для выделения числителя и знаменателя рациональной дроби можно использовать команды стандартной библиотеки numer и denom. Если целое число (знаменатель, числитель) не помещается в строке вывода, то оно переносится на следующую строку, а в качестве знака переноса ставится обратный слэш (\). Maple предоставляет возможность работы с целыми числами, состоящими из более чем пятисот тысяч цифр, но при операциях с такими числами на персональном компьютере, скорее всего, возникнут проблемы с машинным временем и памятью.

Приведем иллюстрации некоторых возможностей пакетов numtheory, GaussInt и padic. Для решения полиномиальных уравнений с рациональными коэффициентами можно использовать команду cfracpol из библиотеки numtheory. В этом случае ответ выводится при помощи цепных дробей; команда cfrac служит для преобразования обычной дроби в цепную и наоборот.

```
> with(numtheory):
Warning, the protected name order has been redefined and unprotected
> a:=cfracpol(68*x^4+364*x^3-281*x^2-91*x+66,20);
```

$a := [-1, 2], [0, 2], [0, 1, 1, 1, 5], [-6]$

```
> b:=cfrac(a[3]); cfrac(b);
```

$$b := \frac{11}{17}$$

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{5}}}}$$

Для демонстрации возможностей библиотеки numtheory при поиске простых чисел воспользуемся следующим тривиальным примером — семизначное число проверяется на простоту и ищется следующее за ним простое число:

```
> d:=7654321: isprime(d); nextprime(d);
```

false

7654337

Поиск ближайшего гауссова числа и проверку на простоту суммы этого числа и мнимой добавки иллюстрирует следующий пример:

```
> with(GaussInt):
Warning, the name GIgcd has been redefined
> g:=GInearest(sqrt(3+I*5)); GIprime(g+I*4);
```

$g := 2 + I$

true

Для p-адических чисел имеется собственное довольно специфическое представление. Для преобразования в данные других типов нужно использовать команду op, которая выводит внутреннюю структуру числа и предоставляет доступ к его операндам.

Приведем пример использования p-адических чисел для действий с экспонентой и логарифмом:

```
> with(padic): Digitsp:=10; evalp(exp(5),5); logp (%);
```

$Digitsp := 10$

$1 + 5 + 3\,5^2 + 3\,5^3 + 4\,5^4 + 5^5 + 2\,5^6 + 4\,5^7 + 3\,5^8 + O(5^9)$

$5 + O(5^{11})$

Статистика

Статистика имеет свою,развитую систему пакетов для решения разнообразных задач. Для пользователей, которые выбрали Maple и нуждаются в средствах статистики, имеется набор команд, позволяющих легко переходить от одной математической специализации к другой, не расходуя лишнего времени на трансформацию данных и освоение различных программных средств.

Пакет stats предоставляет хороший набор команд для анализа данных с вычислением различных средних и квантилей, графического представления данных в виде гистограмм и графиков рассеяния, а также для обработки данных. Команды пакета stats объединены в библиотеки анализа данных (describe), сглаживания (fit), преобразования данных (transform), генерации случайных чисел (random), численной оценки статистических распределений (statevalf), графики (statplots) и анализа вариаций (anova). Также имеются команды считывания данных из файла importdata.

Для работы с командами пакета stats можно подключить весь пакет статистики:

```
> with(stats);
```

[*anova*, *describe*, *fit*, *importdata*, *random*, *statevalf*, *statplots*, *transform*]

В отличие от других пакетов здесь перечислены вложенные в пакет библиотеки. К команде command из библиотеки subpackage можно обращаться следующим образом:

```
subpackage[command](args)
```

Все команды библиотеки subpackage можно загрузить при помощи команды

```
with(stats,subpackage)
```

Чтобы использовать конкретную команду command библиотеки subpackage из пакета stats можно использовать полный вызов:

```
stats[subpackage.command](args)
```

Для получения информации о командах библиотеки subpackage следует воспользоваться справкой

```
?stats[subpackage]
```

или посмотреть соответствующую тему в Help Browser. Там же можно получить сведения о представлении данных (data) и об имеющихся статистических распределениях (distributions).

В пакете реализовано много статистических распределений, как дискретных, так и непрерывных. Пользователям, знакомым с математической статистикой, их назначение ясно из названия команд. Приведем перечень дискретных распределений:

```
binomiald   discreteuniform   empirical
hypergeometric   negativebinomial   poisson
```

Список непрерывных распределений включает:

```
beta      cauchy     chisquare     exponential     fratio
gamma     laplaced     logistic     weibull     - lognormal
normald     studentst     uniform
```

Пакет stats работает с данными, организованными в статистический список. Это может быть комбинация обычных списков Maple (переменная типа list), диагональных матриц и так называемых взвешенных величин. Для взвешенных величин применяется команда Weight(x,n), которая задает последовательность из n чисел величины x. При задании статистического списка можно использовать диапазон значений, оформленный стандартным образом (от..до). Диапазон в этом случае означает величину из него. Для указания отсутствующих данных используется описатель missing. Приведем пример задания статистического списка, подключения библиотеки пакета и применения некоторых команд для определения интервала изменения данных, вариации и среднего значения:

```
> sl:=[6,Weight(0,3),2..4,missing,Weight(-2..5,2)];
```

sl := [6, Weight(0, 3), 2 .. 4, *missing*, Weight(-2 .. 5, 2)]

```
> describe[range](sl);
```

-2 .. 6

```
> with(describe);
```

[*coefficientofvariation*, *count*, *countmissing*, *covariance*, *decile*, *geometricmean*, *harmonicmean*, *kurtosis*, *linearcorrelation*, *mean*, *meandeviation*, *median*, *mode*, *moment*, *percentile*, *quadraticmean*, *quantile*, *quartile*, *range*, *skewness*, *standarddeviation*, *sumdata*, *variance*]

```
> variance(sl);
```

$$\frac{405}{98}$$

```
> evalf(mean(sl));
```

1.714285714

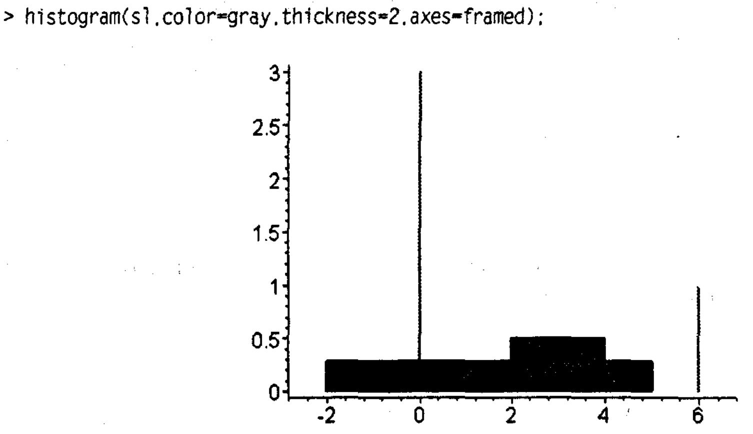
Только пакет describe состоит из 23 команд. Мы не будем перечислять все команды пакета stats, так как сами их имена и справка пакета, по нашему мнению, позволяют пользователю разобраться в работе с любой из них. Отметим, что часть команд пакета stats рассматривалась в других главах и разделах, так, команда leastsquare приближения данных методом наименьших квадратов описана в разделе, посвященном аппроксимации данных.

Продолжим представление команд пакета stats для работы с данными. Воспользуемся возможностями графической библиотеки statplots для построения гистограммы, изображающей набор данных s1:

```
> with(statplots);
```

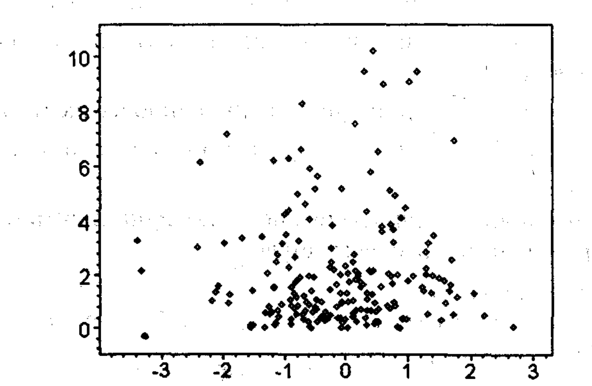
[*boxplot*, *histogram*, *scatterplot*, *xscale*, *xshift*, *xyexchange*, *xzexchange*, *yscale*, *yshift*, *yzexchange*, *zscale*, *zshift*]

```
> histogram(s1,color=gray,thickness=2,axes=framed);
```



Создадим еще два набора данных, первый из которых состоит из двухсот нормально-распределенных случайных чисел, а второй набор подчинен гамма-распределению. После этого для изображения задаваемых этими наборами данных воспользуемся командой scatterplot:

```
> s11:=[random[gamma[1,2]](200)]:
> s12:=[stats[random, normald](200)]:
> statplots[scatterplot](s11,s12,axes=boxed,color=black);
```



Отсортируем оба случайных набора чисел, а затем вычислим коэффициент линейной корреляции между ними:

```
> sl1:=transform[statsort](sl1):
  sl2:=transform[statsort](sl2):
> describe[linearcorrelation](sl1, sl2);
```

.8934095823

Линейная оптимизация

Пакет simplex содержит команды для решения задач линейной оптимизации при помощи симплекс-метода. Перед обращением к командам пакета его нужно подключить или использовать вызов команды с префиксом пакета.

Для определения максимума линейной функции f при ограничениях h применяется команда maximize(f,h). Для поиска минимума используется команда minimize. При использовании пакета simplex эти команды замещают стандартные процедуры из ядра Maple. Если применение команд maximize или minimize не привело к решению, то команда feasible проверит непротиворечивость системы ограничений. Другие команды позволяют выполнять операции, реализующие отдельные шаги симплекс-метода. Перечислим их:

- setup — задание системы линейных уравнений для последующего определения базиса (перечня переменных) при помощи команды basis;
- convexhull — вычисление выпуклой оболочки для набора точек;
- cterm — определение констант для системы уравнений или неравенств;
- define_zero — определение наименьшего ненулевого значения (по умолчанию это значение связано с константой Digits);
- display — вывод заданных линейных уравнений и неравенств в матричной форме;
- dual — вывод сопряженной задачи;
- pivot — конструирование новой системы уравнений с заданным главным элементом;
- pivoteqn — вывод подсистемы для заданного главного элемента;
- pivotvar — вывод переменных, имеющих положительные коэффициенты в выражении целевой функции;
- ratio — вывод отношений для определения наиболее жесткого ограничения;
- standardize — приведение системы уравнений и неравенств к стандартной форме (в виде неравенств).

Приведем простой пример. Определим целевую функцию трех переменных, введем систему ограничений и распечатаем ее:

```
> with(simplex): obj:=-x+2*y+3*z:
Warning, the protected names maximize and minimize have been redefined and unprotected
```

$obj := -x + 2\,y + 3\,z$

```
> cns:={x+2*y-3*z<=4,5*x-6*y+7*z<=8,9*x+10*z<=11}:
> display(cns);
```

$$\begin{bmatrix} 7 & 5 & -6 \\ -3 & 1 & 2 \\ 10 & 9 & 0 \end{bmatrix} \begin{bmatrix} z \\ x \\ y \end{bmatrix} \leq \begin{bmatrix} 8 \\ 4 \\ 11 \end{bmatrix}$$

Попробуем найти максимум функции при наложенных ограничениях:

```
> maximize(obj,cns union {x+y>=0});
```

Максимизация линейной функции трех переменных при четырех ограничениях, наложенных на переменные, к решению не привела. Обращение к команде минимизации позволяет получить решение:

```
> feasible(cns union {x+y>=0});
```

true

```
> minimize(obj,cns union {x+y>=0});
```

$\{z = -2, x = 2, y = -2\}$

Теория графов

Для работы с графами предназначен пакет networks. Граф задается при помощи команд new, complete, cycle или petersen и состоит из вершин и ребер (простых, кратных и петель). Граф представляется в виде процедуры типа GRAPH, тело которой обычно не выводится, так как по умолчанию задан режим подавления вывода interface(verboseproc=0).

Для получения копии графа имеется команда duplicate, таким образом последующие модификации не затрагивают оригинала. Созданный граф может быть изменен при помощи различных команд: для добавления вершин и ребер служат команды addvertex и addedge соответственно, для удаления — delete. По умолчанию вес вершины принят равным нулю, а имена вершин задаются числами, но разрешены любые допустимые в Maple имена. Имена ребер по умолчанию даются в формате e||(1..m) (то есть e1, e2 и т. д.), также допускаются любые, начинающиеся с буквы «e» имена. По умолчанию вес ребра равен единице, но может быть задан Maple-выражением.

Кроме того, существует много команд, реализующих основные операции работы с графами: вычисление потоков в сетях, определение связности, поиск покрывающих деревьев, расчет всех кратчайших путей и т. д. Для изображения графа используется команда draw.

Для генерации случайного графа служит команда random. Для определения характеристик графа предназначены команды edges и vertices. Для получения полного списка команд пакета отошлем к справке Maple, а здесь приведем ряд простых примеров. Сначала подключим пакет networks и создадим пустой граф с именем G:

```
> with(networks):
> G:=new():
```

Добавим к графу пять вершин, а затем свяжем их циклически ребрами:

```
> addvertex({A1,A2,A3,A4,A5},G);
```

A2, A3, A4, A5, A1

```
> addedge(Cycle(A1,A2,A3,A4),G);
```

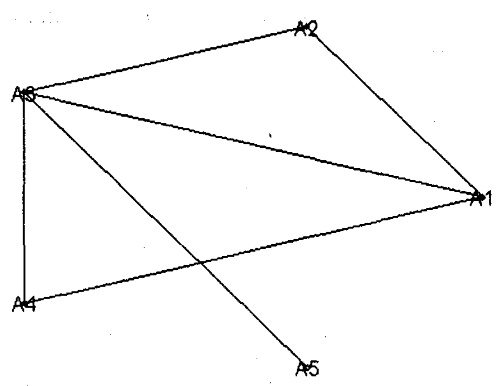
e1, e2, e3, e4

Добавим еще несколько ребер и выведем граф при помощи команды draw:

```
> addedge([{A3,A5},{A1,A3}],G);
```

e5, e6

```
> draw(G);
```



Вычислим максимальную и минимальную степени для вершин графа G и выведем всю имеющуюся информацию о нем:

```
> mindegree(G); maxdegree(G);
```

1

4

```
> show(G);
```

table([*_EdgeIndex* = table(*symmetric*, [(*A2*, *A3*) = { *e2* }, (*A2*, *A1*) = { *e1* },
(*A3*, *A5*) = { *e5* }, (*A3*, *A1*) = { *e6* }, (*A3*, *A4*) = { *e3* },
(*A4*, *A1*) = { *e4* }
]), *_Edges* = { *e1*, *e2*, *e3*, *e4*, *e5*, *e6* }, *_Neighbors* = table([*A2* = { *A3*, *A1* },
A3 = { *A2*, *A4*, *A5*, *A1* }, *A1* = { *A2*, *A3*, *A4* }, *A4* = { *A3*, *A1* },

A5 = { *A3* }
]), *_Head* = table([]), *_Countcuts* = *_Countcuts*, *_Tail* = table([]),
_Bicomponents = *_Bicomponents*, *_Vertices* = { *A2*, *A3*, *A4*, *A5*, *A1* },
_Eweight = table([*e4* = 1, *e5* = 1, *e3* = 1, *e2* = 1, *e1* = 1, *e6* = 1]), *_Ends* = table([
e4 = { *A4*, *A1* }, *e5* = { *A3*, *A5* }, *e3* = { *A3*, *A4* }, *e2* = { *A2*, *A3* }, *e1* = { *A2*, *A1* },

e6 = { *A3*, *A1* }
]), *_Counttrees* = *_Counttrees*, *_Vweight* = table(*sparse*, []),
_Econnectivity = *_Econnectivity*,
_Emaxname = 6
])

В качестве нового объекта определим граф Петерсена. Затем удалим несколько ребер и выведем характеристики модифицированного графа. После этого посчитаем связность и число возможных разрезов:

```
> H:=petersen():
> delete({e||(1,2,3,5,11)},H):
> vertices(H); edges(H); ends(H);
```

{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }

{ *e4*, *e6*, *e12*, *e13*, *e14*, *e15*, *e7*, *e8*, *e9*, *e10* }

{ { 9, 10 }, { 4, 5 }, { 1, 6 }, { 2, 8 }, { 5, 9 }, { 3, 10 }, { 6, 7 }, { 7, 8 }, { 8, 9 }, { 6, 10 } }

```
> connectivity(H); countcuts(H);
```

1

5

Наконец, добавим к графу H направленное ребро, вычислим поток из одной вершины в другую и нарисуем сам граф:

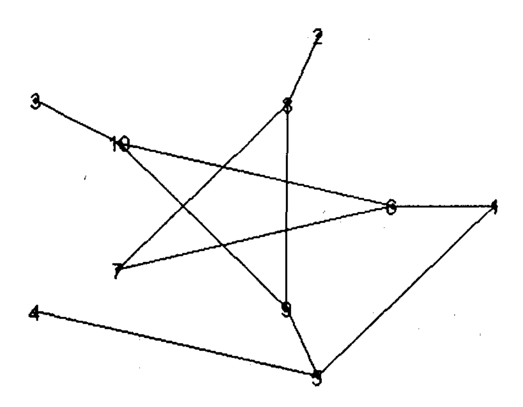
```
> connect(1,5,'weights'=10,'directed',H);
```

e16

```
> flow(H,1,5);
```

11

```
> draw(H);
```



Комбинаторика

Элементарные комбинаторные операции реализуются при помощи команд стандартной библиотеки. Так, при помощи команды binomial(n,r) вычисляются биномиальные коэффициенты. Если n и r являются целыми положительными числами, то это число выборов r объектов из n объектов: binomial(n, r) = n!/r!/(n-r)!. Для всех других случаев используется формула

```
binomial(n, r) = limit(GAMMA(N+1)/GAMMA(R+1)/GAMMA(N-R+1),R=r,N=n)
```

Если в качестве аргументов заданы символьные переменные, то, за исключением простых случаев типа binomial(n,1)=n, возвращается неоцененное выражение. Приведем простые примеры:

```
> binomial(5,2);
```

10

```
> binomial(8, 1/2);
```

$$\frac{65536}{6435}\frac{1}{\pi}$$

```
> binomial(n,5);
```

binomial(n, 5)

```
> expand(%);
```

$$\frac{1}{120}n^5 - \frac{1}{12}n^4 + \frac{7}{24}n^3 - \frac{5}{12}n^2 + \frac{1}{5}n$$

```
> expand(binomial(n,m));
```

binomial(n, m)

Другие операции могут быть выполнены при помощи команд пакетов комбинаторики combinat и комбинаторных структур combstruct. Перечислим некоторые команды пакета combinat. Для вычисления сочетаний и числа сочетаний применяются команды choose и numbcomb, а для вычисления перестановок и их числа имеются команды permute и numbperm соответственно; команда fibonacci служит для определения n-го числа Фибоначчи.

Приведем пример использования перечисленных команд:

```
> restart: with(combinat):
Warning, the protected name Chi has been redefined and unprotected
> choose([a,b,c]); numbcomb([a,b,c]);
```

[[ ], [a], [b], [a, b], [c], [a, c], [b, c], [a, b, c]]

8

```
> permute([a,b,c]); numbperm([a,b,c]);
```

[[a, b, c], [a, c, b], [b, a, c], [b, c, a], [c, a, b], [c, b, a]]

6

Для создания случайно однородных объектов, принадлежащих к одному комбинаторному классу, используется пакет combstruct. Например, для получения всех перестановок некоторого списка достаточно выполнить:

```
> restart: with(combstruct):
> allstructs(Permutation([x,y,z]));
```

[[x, y, z], [x, z, y], [y, x, z], [y, z, x], [z, x, y], [z, y, x]]

В пакете group собраны команды для работы с группами перестановок и конечно-приводимыми группами. Список команд вместе с описанием и примерами можно посмотреть в справке Maple, мы приведем простой пример. Подключим пакет group и создадим группу перестановок:

```
> with(group):
  pg := permgroup(7, {[[1,2,3]],[[1,2,3,4,5,7]]});
```

pg := permgroup(7, { [[1, 2, 3]], [[1, 2, 3, 4, 5, 7]] })

Проверим на принадлежность группе несколько перестановок:

```
> groupmember([[1,2,3,7]],pg);
```

true

```
> groupmember([[1,5],[3,6]], pg);
```

false

Вычислим порядок определенной группы:

```
> grouporder(pg);
```

720

Базис Гребнера

Пакет grobner предназначен для работы с многочленами. Базис Гребнера — это набор многочленов, эквивалентный исходному множеству многочленов. Корни многочленов базиса Гребнера совпадают с корнями исходного множества многочленов. При этом коэффициенты многочленов базиса составляют верхнюю треугольную матрицу. В состав пакета входят следующие команды: finduni, finite, gbasis, gsolve, leadmon, normalf, solvable, spoly.

Рассмотрим пример. Сначала подключим пакет и зададим множество из трех полиномов относительно переменных x, y и z:

```
> with(grobner, gbasis):
> P:=[z^2-2*x*y*z+5*x, x*y^2+y*z^3+2, 3*x*y^2-8*z^3];
```

$$P := [z^2 - 2\,y\,z\,x + 5\,x,\, x\,y^2 + y\,z^3 + 2,\, 3\,x\,y^2 - 8\,z^3]$$

```
> G:=gbasis(P,[x,y,z],plex);
```

$$G := [75\,x - 32\,z^5 - 16\,z^3 + 15\,z^2 - 12,\, -60\,z^4 + 9\,y - 64\,z^5 - 48\,z^2 - 32\,z^3 - 24,$$
$$24\,z^3 + 9 + 30\,z^7 + 32\,z^8 + 24\,z^5 + 16\,z^6]$$

Обратим внимание, что третий многочлен G[3] зависит только от переменной z. Найдем численно его корни:

```
> so:=fsolve(G[3],{z},complex):
```

Теперь можно найти все корни системы нелинейных уравнений:

```
> for sz in so do
sy:=solve(subs(sz,G[2]),{y});
sx:=solve(subs(sy,sz,G[1]),{x});
print(sx[],sy[],sz[])
od:
```

$x = -1.92221,\, y = -1.66084,\, z = -1.25752$

$x = -.0404813,\, y = 4.33716,\, z = -.658514$

$x = .0918000 + .0928427\,I,\, y = -4.46789 + .820911\,I,\, z = -.373182 - .931470\,I$

$x = .0918000 - .0928427\,I,\, y = -4.46789 - .820911\,I,\, z = -.373182 + .931470\,I$

$x = .284283 + .169908\,I,\, y = .951578 + 1.75980\,I,\, z = .288450 - .737768\,I$

$x = .284283 - .169908\,I,\, y = .951578 - 1.75980\,I,\, z = .288450 + .737768\,I$

$x = .0113385 + .0481443\,I,\, y = -.488446 - 4.58358\,I,\, z = .574001 - .456134\,I$

$x = .0113385 - .0481443\,I,\, y = -.488446 + 4.58358\,I,\, z = .574001 + .456134\,I$

Для проверки подставим последнее решение в исходное множество многочленов:

```
> subs(sx,sy,sz,P);
```

$[-.15\ 10^{-9} - .2\ 10^{-9}\ I, .1\ 10^{-8} - .10\ 10^{-8}\ I, 0. + 0.\ I]$

Видно, что найденные решения являются корнями трех исходных полиномов с погрешностью округлений.

Алгебры и формы

В данном разделе кратко обсудим ряд пакетов, предназначенных для проведения абстрактных математических выкладок. Пакет difforms предоставляет команды для создания дифференциальных форм и работы с ними. Пакет работает с формами (тип form), в формировании которых участвуют величины скалярного типа (scalar) и константы (const). Например, при помощи команды defform можно определить базис, а для преобразований использовать операции внешнего умножения (&^) и производной (d).

Для получения сведений об используемых пакетом типах следует обратиться к справке ?difforms[type], где в качестве type могут выступать const, scalar или form. Приведем пример:

```
> with(difforms): defform(f=scalar, g=scalar, d(l)=e);
> type(f*g,form);
```

false

```
> f*(&^(u,v)+&^(u,w))+g*&^(u,v);
  simpform(%);
```

$f((u \&\^{} v) + (u \&\^{} w)) + g\,(u \&\^{} v)$

$(f+g)\,(u \&\^{} v) + f\,(u \&\^{} w)$

```
> type(f,scalar);
```

true

```
> d(l); d(l^2); d(a);
```

e

$(1+(-1)^{\mathrm{wdegree}(l)})\,(e \&\^{} l)$

d(a)

В пакете liesymm, позволяющем работать с симметриями Ли, используются свои внешняя производная (d) и внешнее произведение (&^), не зависимые от имеющихся в пакете difforms. Определение списка координатных переменных (нуль-форм) производится при помощи команды setup, для вычисления производной Ли служит команда Lie, команда wcollect применяется для представления формы в виде суммы форм. Для преобразования форм и представления результатов имеются команды choose, getcoeff, mixpar, wdegree, wedgeset и value.

Команды пакета не требуют непосредственной работы с дифференциальными формами. Если задан набор дифференциальных уравнений, то при помощи команды determine определяются их координаты и дифференциальные формы. Для обозначения частных производных лучше использовать команду Diff, чем команды diff и D.

Для определения нужного порядка смешанных производных можно использовать команду mixpar, а для конвертирования производных Diff в производные diff при представлении результатов работы команды determine нужно использовать команду value. Проиллюстрируем работу некоторых команд.

Напомним, что идея применения групп Ли [64] для анализа дифференциальных уравнений заключается в поиске полной группы симметрий системы дифференциальных уравнений, что позволяет строить новые решения системы по уже известным. В качестве примера рассмотрим хорошо известное уравнение Кортевега–де Фриза. Анализ существующих у данного уравнения непрерывных симметрий можно найти в книге [64], а их поиск средствами Maple обсуждался в книге [8]. Следуя [8, 64], найдем с помощью команд пакета liesymm симметрии уравнения Кортевега–де Фриза. Сначала подключим пакет и определим дифференциальное уравнение:

```
> with(liesymm):
Warning, the protected name close has been redefined and unprotected

> eq:=Diff(u(x,t),t)+u(x,t)*Diff(u(x,t),x)-
   Diff(u(x,t),x,x,x)=0;
```

$$eq := \left(\frac{\partial}{\partial t}\,\mathrm{u}(x,t)\right) + \mathrm{u}(x,t)\left(\frac{\partial}{\partial x}\,\mathrm{u}(x,t)\right) - \left(\frac{\partial^3}{\partial x^3}\,\mathrm{u}(x,t)\right) = 0$$

Теперь вычислим генераторы группы дифференциального уравнения, а затем упростим их:

```
> eqs := autosimp(determine(eq,V,u(x,t),w));
```

$$eqs := \{\ \}\ \&\text{where}\ \{\mathrm{V3_4}(t) = C1,\ \mathrm{V3_5}(t) = C2,\ \mathrm{V3_2}(x,t) = x\,C1 + C2,$$
$$\mathrm{V1_2}(t) = t\,C1 + C6,\ \mathrm{V3_3}(t) = -t\,C1 + C8,\ \mathrm{V1_3}(t) = t\,C2 + C4,$$
$$\mathrm{V1}(x,t,u) = x\,(t\,C1 + C6) + t\,C2 + C4,\ \mathrm{V1_1}(x,t) = x\,(t\,C1 + C6) + t\,C2 + C4,$$
$$\mathrm{V3}(x,t,u) = u\,(-t\,C1 + C8) + x\,C1 + C2,\ -t\,C1 - C8 - 2\,C6 = 0,$$
$$\mathrm{V3_1}(x,t) = -t\,C1 + C8,\ \mathrm{V2_2}(t) = C9 + \frac{3}{2}\,t^2\,C1 + 3\,C6\,t,$$
$$\mathrm{V2}(x,t,u) = C9 + \frac{3}{2}\,t^2\,C1 + 3\,C6\,t\}$$

Заметим, что среди полученных равенств десятое выполняется только в случае C1=0 и C8=-2C6. Подставим эти условия в генераторы группы:

```
> subs(C1=0,C8=-2*C6,op(2,eqs));
```

$$\{\mathrm{V3_5}(t) = C2,\ \mathrm{V1_3}(t) = t\,C2 + C4,\ 0 = 0,\ \mathrm{V3_3}(t) = -2\,C6,$$
$$\mathrm{V3}(x,t,u) = -2\,u\,C6 + C2,\ \mathrm{V3_1}(x,t) = -2\,C6,\ \mathrm{V3_4}(t) = 0,\ \mathrm{V3_2}(x,t) = C2,$$
$$\mathrm{V1_2}(t) = C6,\ \mathrm{V1}(x,t,u) = x\,C6 + t\,C2 + C4,\ \mathrm{V1_1}(x,t) = x\,C6 + t\,C2 + C4,$$
$$\mathrm{V2_2}(t) = C9 + 3\,C6\,t,\ \mathrm{V2}(x,t,u) = C9 + 3\,C6\,t\}$$

Следующей командой выделим условия, при которых векторное поле

$$v = \mathrm{V1}(x,t,u)\,\partial_x + \mathrm{V2}(x,t,u)\,\partial_t + \mathrm{V3}(x,t,u)\,\partial_u$$

порождает группу симметрий:

```
> select(has,%,{V1,V2,V3});
```

$\{V3(x, t, u) = -2\,u\,C6 + C2,\ V1(x, t, u) = x\,C6 + t\,C2 + C4,\ V2(x, t, u) = C9 + 3\,C6\,t\}$

В полученных выражениях C2, C4, C6, C9 — произвольные постоянные. Таким образом, алгебра симметрий уравнения Кортевега–де Фриза порождается четырьмя векторными полями: $v_1 = \partial_x$ — сдвиг в пространстве, $v_2 = \partial_t$ — сдвиг во времени, $v_3 = t\partial_x$ — преобразование Галилея и $v_4 = x\partial_x + 3t\partial_t - 2u\partial_u$ — растяжение.

В пакете diffalg собраны команды, реализующие алгебраические операции над полем полиномиальных дифференциальных уравнений. Пакет включает более двадцати команд, позволяющих оперировать с такими дифференциальными уравнениями, в частности понижать их порядок, разыскивать решение в виде формального ряда. Для представления уравнений в пакете предусмотрена сокращенная форма записи, так называемое jet-представление. Для подробной информации об этом пакете не обойтись без справки Maple, а мы ограничимся небольшим примером. Подключим пакет diffalg, определим алгебраическое кольцо и преобразуем в jet-представление уравнение Кортевега–де Фриза:

```
> with(diffalg):
> R := differential_ring(ranking=[u], derivations=[x,t],
                          notation=diff);
```

$R := PDE_ring$

```
> p:=denote(eq, "jet", R);
```

$p := u_t + u_{[\ ]}\,u_x - u_{x,x,x}$

Теперь выведем присутствующие в уравнении производные в порядке их возрастания:

```
> derivatives(p,R,'increasingly');
```

$$\left[u(x,t),\ \frac{\partial}{\partial t}u(x,t),\ \frac{\partial}{\partial x}u(x,t),\ \frac{\partial^3}{\partial x^3}u(x,t)\right]$$

Еще один алгебраический пакет Ore_algebra является коллекцией команд, реализующих операции алгебры линейных полиномиальных, дифференциальных и разностных операторов. Команды пакета можно условно разделить на три группы:

- команды создания алгебр;
- команды для вычисления в созданных алгебрах;
- команды преобразования выражений.

При создании алгебр указываются объекты и их свойства. Проиллюстрируем ряд команд пакета примерами. Подключим пакет и создадим полиномиальную алгебру, наложив одно условие на объекты, а затем вычислим скалярное произведение двух выражений;

```
> with(Ore_algebra):
> B:=poly_algebra(a,b,x,y,alg_relations={a*b+1});
```

$B := Ore_algebra$

```
> skew_product(x+b,y+a,B);
```

$a\,x + x\,y + b\,y - 1$

Теперь определим алгебру дифференциальных операторов, зададим выражение и преобразуем его к стандартному для Maple виду:

```
> A:=diff_algebra([Dx,x],[Dy,y]);
```

$A := Ore_algebra$

```
> P:=Dx^2+Dy^2-x*y;
 Ore_to_diff(P,f,A);
```

$P := Dx^2 + Dy^2 - x\,y$

$$-x\,y\,\mathrm{f}(x,y) + \left(\frac{\partial^2}{\partial x^2}\mathrm{f}(x,y)\right) + \left(\frac{\partial^2}{\partial y^2}\mathrm{f}(x,y)\right)$$

Пакет GF предназначен для определения конечного поля Галуа и операций над его константами и функциями. Пакет не требует дополнительного подключения. Приведем пример определения поля (команда GF) и нескольких операций:

```
> G1 := GF(2,3,alpha^3-alpha+1):
> a := G1[ConvertIn](alpha);
```

$a := \alpha$

```
> c:=G1["^"](a,5);
```

$c := 1 + \alpha + \alpha^2$

```
> G1["+"](a,c);
```

$1 + \alpha^2$

Для работы с алгебраическими кривыми предназначен пакет algcurves. В состав пакета входят команды позволяющие преобразовывать алгебраические кривые, вводить на них параметризацию, вычислять монодромию и многое другое. Приведем простой пример. Подключим пакет, определим алгебраическую кривую, затем введем на ней параметризацию и вычислим монодромию:

```
> with(algcurves):
> f:=x^2+y^2-1:
> parametrization(f,x,y,t);
```

$$\left[2\frac{t}{1+t^2}, \frac{-1+t^2}{1+t^2}\right]$$

```
> monodromy(f,x,y);
```

[-1.8000000000, [-1.49666295471 *I*, 1.49666295471 *I*],
[[-1., [[1, 2]]], [1., [[1, 2]]]]]

Пакет Domains (старое название — Gauss) является инструментальным средством для реализации сложных алгоритмов. Разработчики Maple считают, что заложенные в этом пакете возможности по конструированию «области вычислений» позволяют писать лучшие коды. Вслед за фирмой Waterloo Maple Inc. мы полагаем, что заинтригованный читатель посмотрит файл справки с соответствующим примером по команде ?Domains,example.



Maple и другие программы

В этой главе рассматривается взаимодействие Maple с языками программирования, текстовыми редакторами, программами среды Microsoft Office и пакетом численного анализа MATLAB. Это необходимо для анализа результатов аналитических выкладок Maple при помощи других программ и их использования при подготовке текстовых материалов (статей, книг, WWW-страниц и др.). Одним из вариантов использования аналитических результатов в численном анализе является их преобразование в коды языков С или Фортран с последующим программированием на них. Другой путь состоит в применении высокоэффективных алгоритмов пакета MATLAB, описанию возможностей которого посвящена вторая часть этой книги. Maple поддерживает перевод рабочих документов в наиболее распространенные форматы для публикаций, такие как RTF, LaTeX и HTML.

Генерация кодов Maple, С и Фортран

Пакет codegen предназначен для оптимизации текстов программ на языке Maple и перевода Maple-выражений в конструкции языков программирования. В этом разделе мы остановимся на некоторых командах работы с Maple-выражениями и перевода их на традиционные языки программирования — С и Фортран. Перед обращением к пакету его нужно подключить командой `with(codegen)`.

Сначала перечислим некоторые команды из пакета codegen, предназначенные для модернизации текстов Maple-процедур и выражений:

- `makeproc(a,x)` — переводит выражение `a` (или переменную типа `list`, состоящую из выражений) в процедуру с параметром `x` (или набором параметров);
- `makeparam(x,f)` — преобразует переменную `x`, входящую в Maple-процедуру `f`, во входной параметр;
- `makeglobal(x,f)` — преобразует переменную `x`, входящую в Maple-процедуру `f`, в глобальную переменную;
- `declare(x::t,f)` — описывает переменную `x`, входящую в Maple-процедуру `f`, как переменную типа `t`;

- makevoid(f) — преобразует Maple-процедуру с именем f в процедуру без возвращаемого значения;
- optimize(expr) — оптимизирует Maple-выражение expr. В качестве expr может выступать алгебраическое выражение, массив, набор уравнений или процедура;
- prep2trans(f) — преобразует процедуру f к виду, удобному для перевода на алгоритмические языки. В частности, кусочно-непрерывныая функция (piecewise) представляется как условный оператор, а сумма (Sum) — как цикл;
- cost(expr) — оценивает число элементарных операций, необходимых для выполнения выражения expr.

Проиллюстрируем действие перечисленных команд на примере.

Подключим пакет codegen:

```
> with(codegen):
```

Переменной f присвоим выражение, зависящее от x и t:

```
> f:=(x-t)^4;
```

$f := (x - t)^4$

Теперь преобразуем алгебраическое выражение f в процедуру ff с одним входным параметром x неопределенного типа:

```
> ff:=makeproc(f,x);
```

ff := **proc** (*x*) (*x* – *t*)^4 **end proc**

Оценим число операций, необходимых для выполнения процедуры ff:

```
> cost(ff);
```

3 *multiplications* + *additions*

Создадим новую процедуру ff1, которая является оптимизированным вариантом процедуры ff:

```
> ff1:=optimize(ff);
```

ff1 := **proc** (*x*) **local** *t2*; *t2* := (*x* – *t*)^2; *t2*^2 **end proc**

Оценим число операций, необходимых для выполнения процедуры ff1:

```
> cost(ff1);
```

storage + *assignments* + 2 *multiplications* + *additions*

Опишем тип входного x параметра как float:

```
> declare(x::float,ff1);
```

proc (*x*::*float*) **local** *t2*; *t2* := (*x* – *t*)^2; *t2*^2 **end proc**

Преобразуем процедуру ff1 в процедуру gg1 без возвращаемого значения:

```
> gg1:=makevoid(ff1);
```

gg1 := **proc** (*x*::*float*) **local** *t2*; *t2* := (*x* – *t*)^2; RETURN( ) **end proc**

Преобразуем переменную t, входящую в тело процедуры, во входной параметр типа float:

```
> gg1:=makeparam(t::float,gg1);
```

gg1 := **proc** (*x*::*float*, *t*::*float*) **local** *t2*; *t2* := (*x* – *t*)^2; RETURN( ) **end proc**

Опишем локальную переменную t2 из тела процедуры как глобальную:

```
> gg1:=makeglobal(t2,gg1);
```

gg1 := **proc** (*x*::*float*, *t*::*float*) **global** *t2*; *t2* := (*x* − *t*)^2; RETURN( ) **end proc**

Для иллюстрации работы команды prep2trans определим функцию f, которая вычисляет интеграл функции sin(ax) на интервале от 0 до π по формуле Симпсона. Для этого воспользуемся командой simpson пакета student и командой makeproc:

```
>f:=makeproc(student[simpson](sin(a*x),x=0..Pi,10),
           a::float);
```

f := **proc** (*a*::*float*)

    1/30×π×(sin(*a*×π) + 4×Sum(sin(1/10×*a*×(2×*i* − 1)×π), *i* = 1 .. 5)

    + 2×Sum(sin(1/5×*a*×*i*×π), *i* = 1 .. 4))

end proc

Теперь обратимся к команде prep2trans:

```
> prep2trans(f);
```

proc (*a*::*float*)

local *i1*, *i2*, *s1*, *s2*, *t1*, *t2*;

    s1 := 0;

    for *i1* **to** 5 **do** *s1* := *s1* + sin(1/10×*a*×(2×*i1* − 1)×π) **end do** ;

    s2 := 0;

    for *i2* **to** 4 **do** *s2* := *s2* + sin(1/5×*a*×*i2*×π) **end do** ;

    1/30×π×(sin(*a*×π) + 4×*s1* + 2×*s2*)

end proc

Обратим внимание на то, что все суммы в результате действия команды prep2trans были преобразованы в операторы циклов.

Описанные процедуры часто используются для приведения Maple-выражений к виду, удобному для преобразования в программы на языках программирования. Перейдем теперь непосредственно к командам перевода Maple-выражений на алгоритмические языки.

Вывод выражения в виде Фортран-операторов осуществляется по команде

```
fortran(expr,options)
```

Здесь в качестве expr могут выступать выражение, массив выражений, процедура или список уравнений. Параметры options могут принимать следующие значения: filename="name.ext" (результат выводится в файл), optimized (Фортран-выражения оптимизируются, вводятся вспомогательные переменные), digits и mode (задание точности вычислений).

Чтобы записать выражение на языке C, необходимо пользоваться командой

```
C(expr,options)
```

Параметры этой команды аналогичны параметрам, описанным для команды fortran.

Приведем примеры использования этих двух команд. Сначала определим выражение f, а затем переведем его на С и Фортран:

```
> f:=a+b^5*sin(b^3);
```

$f := a + b^5 \sin(b^3)$

```
> C(f);
      t0 = a+b*b*b*b*b*sin(b*b*b);
> fortran(f);
      t0 = a+b**5*sin(b**3)
```

Теперь обратимся к команде C с параметром оптимизации текста выражения:

```
> C(f,optimized);
      t1 = b*b;
      t2 = t1*t1;
      t5 = sin(t1*b);
      t7 = a+t2*b*t5;
```

Для оптимизации выражения введены дополнительные переменные, которые необходимо описывать вручную в программе на С. Это не так сложно в случае небольших формул, но может стать очень трудоемким для громоздких выражений. Для избежания ручного труда преобразуем выражение в процедуру с двумя параметрами и снова обратимся к команде C:

$g := \mathbf{proc}(a{::}\mathit{float}, b{::}\mathit{float})\ a + b\text{^}5\times\sin(b\text{^}3)\ \mathbf{end\ proc}$

```
> C(g,optimized);
#include <math.h>
double g(a,b)
double a;
double b;
{
  double t1;
  double t2;
  double t5;
  {
    t1 = b*b;
    t2 = t1*t1;
    t5 = sin(t1*b);
    return(a+t2*b*t5);
  }
}
```

В результате получился текст процедуры на языке С, не требующий никакой дополнительной работы. Приведем пример использования команды генерации кодов языка Фортран с сохранением текста результата в файл:

```
> fortran(g,optimized,filename="res.for");
```

В результате выполнения команды будет создан файл res.for со следующим текстом:

```
      real function g(a,b)
      real a
      real b
```

```
real t1
real t2
real t5
  t1 = b**2
  t2 = t1**2
  t5 = sin(t1*b)
  g = a+t2*b*t5
  return
end
```

В Maple 6.0 появилась возможность подключения внешних процедур, написанных на алгоритмических языках. Для описания и вызова внешних процедур используются соответственно команды define_external и call_extarnal. В документации по пакету и в справочной системе дано описание этих команд и приведен пример процедуры. При вызове внешних процедур важно согласование типов данных Maple и языков программирования, соответствие которых изложено в руководстве по пакету. Отметим, что по умолчанию Maple использует транслятор Microsoft C для операционной системы Windows. Таким образом, процедуры, написанные на C (или другом языке, поддерживающем интерфейс C), и находящиеся в библиотеках (DLL в среде Windows), могут теперь подключаться к Maple динамически и вызываться как команды Maple.

Преобразование документов Maple в форматы LaTeX, RTF, HTML

В этом разделе мы рассмотрим взаимодействие Maple с системами подготовки научных публикаций, как традиционных, так и электронных. Замечательная способность Maple представлять математические результаты в форматах, стандартных для систем публикаций, используется уже давно. Maple позволяет автоматически переводить рисунки и формулы, полученные в результате громоздких вычислений, в нужный формат без дополнительной ручной обработки. В настоящее время система верстки LaTeX фактически является стандартом для научных публикаций. Описанию работы с ней посвящена глава 19 «Краткое введение в пакет LaTeX». Здесь мы только упомянем, что в LaTeX существует свой язык для написания формул.

Для перевода математического выражения expr в формат системы редактирования LaTeX в Maple используется команда

```
latex(expr)
```

или

```
latex(expr,"name.ext")
```

Во втором варианте результат работы команды выводится в файл. Полученные выражения можно легко вставлять в LaTeX-документы, но следует помнить, что полученную формулу необходимо включить в нужное окружение (например, в символы $ для формул внутри текста или конструкции \begin{equation} ... \end{equation} для выделенных в строку формул). Приведем пример обращения к команде latex:

```
> Int(1/(x^2+1), x) = int(1/(x^2+1), x);
```

$$\int \frac{1}{x^2+1}\,dx = \arctan(x)$$

```
> latex(%);
\int \!\left ({x}^{2}+1\right )^{-1}{dx}=\arctan(x)
```

В LaTeX-документе эта формула при ее размещении в отдельной строке должна выглядеть следующим образом:

```
\begin{equation}
\int \!\left ({x}^{2}+1\right )^{-1}{dx}=\arctan(x)
\end{equation}
```

Обратим внимание на то, что результат преобразования выражения отличается от вида выражения в Maple-документе, так как здесь используется отрицательная степень, а не дробь. Таким образом, любое математическое выражение можно преобразовать в формат LaTeX.

В Maple существует возможность перевода в различные форматы всего Maple-документа. Преобразовать рабочий документ можно из основного меню Maple, выбрав пункт File, а затем Export As или Save As. После этого надо выбрать требуемый формат документа при помощи дополнительного меню. Список возможных форматов дан в табл. 9.1.

Таблица 9.1. Типы файлов, в которые может быть преобразован рабочий документ

Тип документа	Описание
Maple text	Текстовый формат с записью математических выражений на языке Maple
Plain text	Текстовый формат с имитацией математических выражений
Maple explorer	Формат, близкий к формату LaTeX
LaTeX	LaTeX-документ
RTF (Rich text format)	Расширенный текстовый формат
HTML	Гипертекстовый формат

Теперь рассмотрим подробнее преобразование Maple-документов в форматы LaTeX, RTF и HTML. На рис. 9.1 приведен короткий Maple-документ с именем latexword.mws, включающий различные объекты. Будем использовать далее этот документ для демонстрации перевода документов в различные форматы.

При преобразовании Maple-документа в LaTeX все рисунки автоматически преобразуются в формат Encapsulate PostScript, а математические выражения переводятся на язык формул пакета LaTeX. Рисунки сохраняются в файлах с именами, состоящими из имени документа и номера рисунка. При записи формул Maple использует свои команды и окружения, которые реализованы в пакете для LaTeX maple2e. Стилевые файлы пакета maple2e находятся в каталоге ETC. Приведем текст, полученный в результате преобразования тестового документа (см. рис. 9.1) в формат LaTeX. В результате конвертации был создан файл latexword.tex и файл latexword01.eps с рисунком. Текст документа LaTeX приводится ниже в сокращенном виде и снабжен нашими комментариями на русском языке:

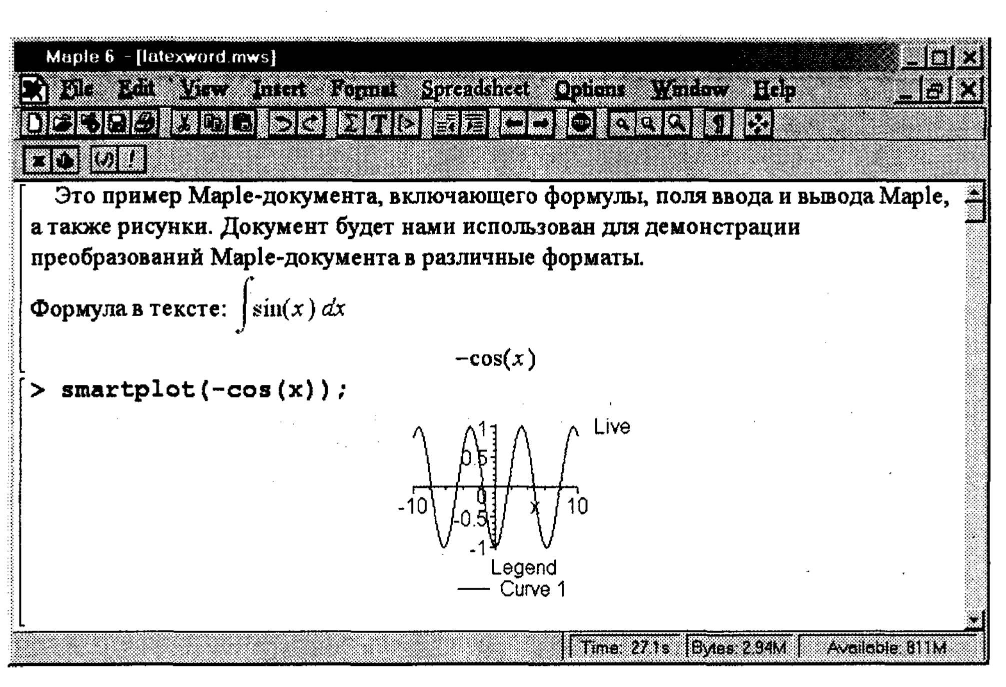


Рис. 9.1. Окно Maple-сессии с примером документа

```
%% Created by Maple 6 (IBM INTEL NT)
%% Source Worksheet: latexword.mws
%% Generated: Tue Oct 10 23:04:48 2000
\documentclass{article}
% Подключение пакета Maple для LaTeX
\usepackage{maple2e}
% Определение команд
\def\emptyline{\vspace{12pt}}
\DefineParaStyle{Maple Output}
\begin{document}
\pagestyle{empty}
% Открытие окружения maplegroup
\begin{maplegroup}
\begin{flushleft}
Это пример Maple-документа, включающего формулы, поля ввода и вывода Maple,
а также рисунки. Документ будет нами использован для демонстрации
преобразований Maple-документа в различные форматы.

Формула в тексте:

\mapleinline{active}{2d}{int(sin(x),x);}{%$\int \mathrm{sin}(x)\,dx$%}
\end{flushleft}
\mapleresult
\begin{maplelatex} \mapleinline{inert}{2d}{-cos(x);}{% \[ - \mathrm{cos}(x) \] %} \end{maplelatex}
\end{maplegroup}
\begin{maplegroup}
\begin{mapleinput}
```

```
\mapleinline{active}{1d}{smartplot(-cos(x));}{%}
\end{mapleinput}
\mapleresult
% Включение рисунка в формате EPS
\begin{center} \mapleplot{latexword01.eps} \end{center}
\end{maplegroup}
\end{document}
%% End of Maple 6 Output
```

В Maple не предусмотрена языковая поддержка LaTeX, значит, для русификации текста необходимо подключить соответствующие библиотеки. Русификация будет зависеть от используемой версии LaTeX. Так, в случае пакета MikTeX и кодировки Windows необходимо после строки с командой `\documentclass` вставить следующие команды:

```
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
```

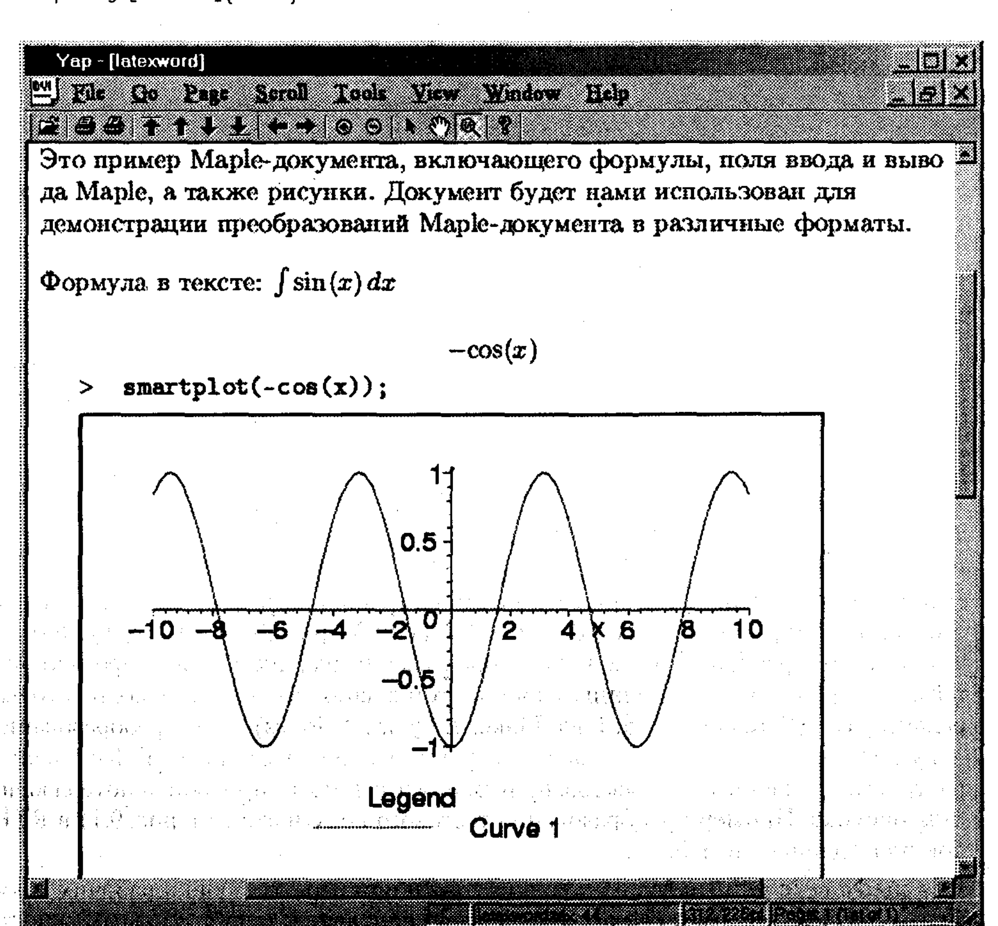


Рис. 9.2. Результат обработки документа LaTeX

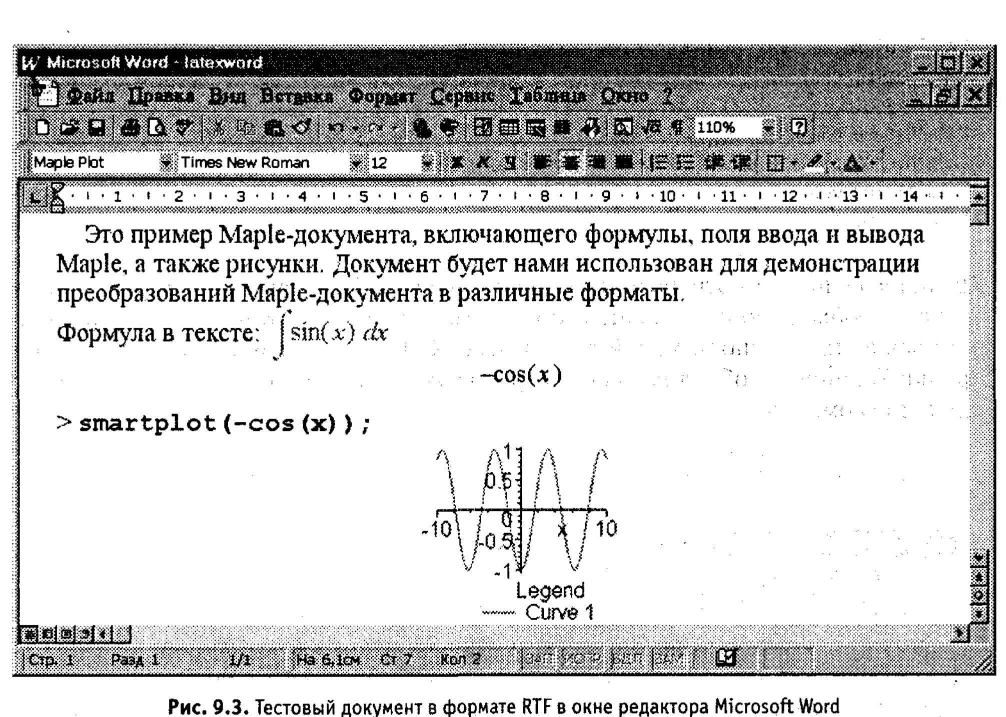


Рис. 9.3. Тестовый документ в формате RTF в окне редактора Microsoft Word

После этого документ может быть обработан транслятором LaTeX. В результате будет получен DVI-файл, который может быть распечатан или выведен на экран. На рис. 9.2 изображено окно программы Yap для просмотра DVI-файлов с открытым файлом latexword.dvi. Отметим, что для правильной работы транслятора файлы пакета maple2e должны находиться в каталоге с транслируемым документом или в каталоге LaTeX со стилевыми файлами.

Изображенный на рис. 9.2 результат трансляции документа LaTeX отличается от исходного документа (рис. 9.1). В частности, рисунок в формате EPS имеет больший размер и иное представление, чем оригинал, что объясняется спецификой работы драйвера Maple преобразования рисунков в формат EPS.

Другим распространенным инструментом подготовки публикаций является текстовый редактор Microsoft Word. В новой версии Maple 6.0 рабочий документ автоматически преобразуется со всеми формулами и графиками в RTF-формат, что позволяет работать с полученным документом в наиболее популярных текстовых редакторах (Microsoft Word, Page Maker и Quark X Press). При преобразовании документа в этот формат все графики и формулы внедряются в RTF-файл в виде растровых графических объектов, причем при этом может произойти потеря качества рисунка. Пример преобразования тестового документа (см. рис. 9.1) в RTF-документ дан на рис. 9.3.

Начиная с версии 5.1 в Maple появилась функция перекодировки рабочих документов в HTML-формат (Hyper Text Markup Language Format). Эта возможность позволяет использовать Maple и как среду автоматической разработки математических электронных публикаций. Поскольку последние версии Microsoft Word поддер-

живают HTML-формат, то это предоставляет еще одну возможность автоматического преобразования Maple-документов в текстовые документы Word.

В результате преобразования по умолчанию создается основной файл с расширением .html и именем, совпадающим с именем Maple-документа, который должен загружаться программами, поддерживающими фреймы, например Netscape или Microsoft Explorer. Кроме того, создается ряд дополнительных HTML-файлов и файлы с рисунками, причем все формулы и рисунки преобразуются в растровые графические файлы формата GIF. Все эти файлы имеют имена с номерами. В основном файле содержатся ссылки на дополнительные HTML-файлы. Перед конвертацией появляется диалоговое окно, в котором предлагается выбрать имя каталога для размещения графических файлов и можно отказаться от использования фреймов. Ниже приведен текст основного HTML-файла с поддержкой фреймов для нашего примера:

```
<html>  <head>
<title>latexword.html</title>
<!- Created by Maple 6, IBM INTEL NT ->
</head>
<basefont size=3> <frameset cols="25%,*">
<frame src="latexwordTOC.html" name="TableOfContents">
<frame src="latexword1.html" name="Content">
<noframes> Sorry, this document requires that your browser support frames.
<a href="latexword1.html" target="Content">This link</a> will take you to a non-frames
presentation of the document. </noframes>
</frameset>
</basefont>
</html>
```

Результат считывания полученного документа программой Microsoft Explorer дан на рис. 9.4.

Взаимодействие с MATLAB

Хотя версия Maple 6.0 уже практически ни в чем не уступает другим пакетам при реализации многих операций численного анализа, иногда удобно воспользоваться для проведения вычислений высокоэффективными алгоритмами MATLAB. Описанию этой системы посвящена вторая часть книги, а здесь мы рассмотрим вызов команд MATLAB из сеанса Maple и обмен данными между двумя вычислительными средами. Понятно, что это обсуждение полезно только в том случае, если на компьютере правильно установлены оба этих программных продукта. Отметим еще, что в состав MATLAB может входить библиотека символьных вычислений из ядра Maple, что также описано во второй части книги.

Возможность обмена информацией между двумя вычислительными средами посредством файлов предоставляют команды ввода-вывода из пакета Maple LinearAlgebra (см. главу 5 «Алгебра в Maple»), которые обеспечивают ASCI стандарт записи информации MATLAB. Например, если в MATLAB была записана матрица A=[1:3;4:6] командой

```
save d:\temp\test -ascii
```

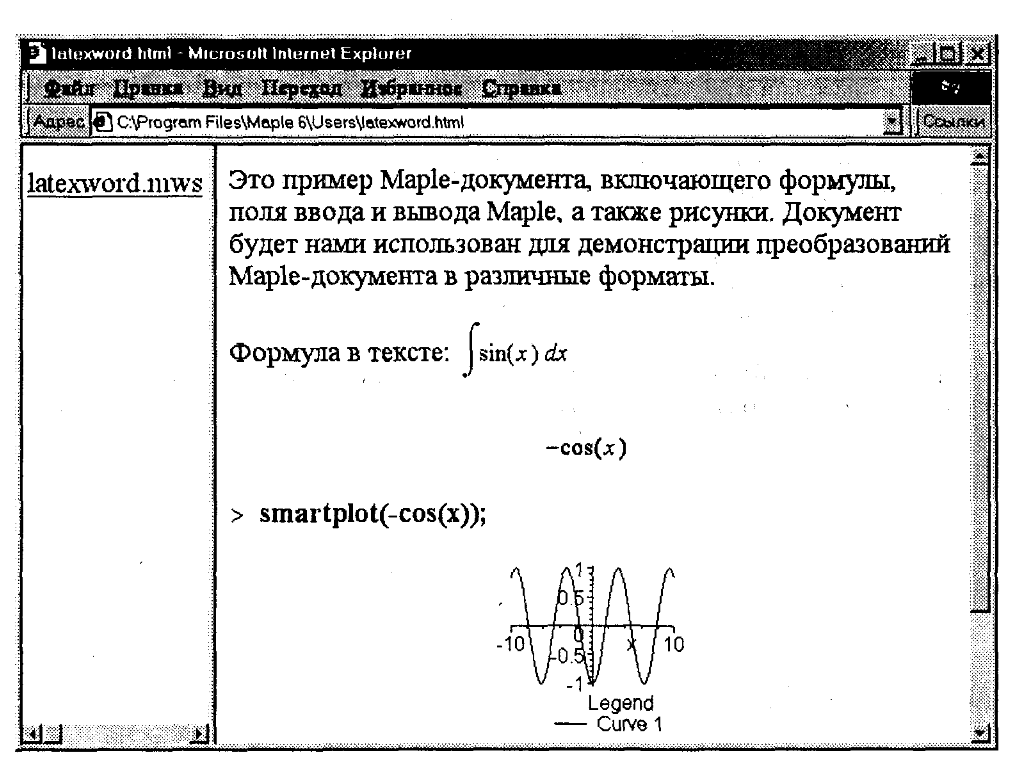


Рис. 9.4. Тестовый документ в формате HTML в окне программы Microsoft Internet Explorer

то считать ее в Maple можно следующим образом:

```
> C:=ImportMatrix("d:\\temp\\test",source=Matlab);
```

$$C := \begin{bmatrix} 1.0000000 & 2.0000000 & 3.0000000 \\ 4.0000000 & 5.0000000 & 6.0000000 \end{bmatrix}$$

Более гибкое и универсальное взаимодействие обеспечивает пакет Matlab, который позволяет осуществлять обмен данными между сеансами двух вычислительных сред и обращаться из Maple к командам MATLAB. Пакет нужно подключить командой

```
> with(Matlab);
```

[*chol*, *closelink*, *defined*, *det*, *dimensions*, *eig*, *evalM*, *fft*, *getvar*, *inv*, *lu*, *ode45*, *openlink*, *qr*, *setvar*, *size*, *square*, *transpose*]

В результате команды из приведенного списка становятся доступны. В случае неправильной установки MATLAB библиотека найдена не будет и вместо приведенного списка появится сообщение об ошибке. При подключении пакета или при обращении к любой из его команд происходит автоматический запуск программы MATLAB (см. рис. 9.5). Перечислим в следующих двух таблицах команды пакета Matlab. В табл. 9.2 даны команды, реализующие известные численные методы, причем эти команды автоматически поддерживают обмен данными между двумя программами. В табл. 9.3 даны команды Maple, обеспечивающие динамическое взаимодействие Maple и MATLAB.

Таблица 9.2. Математические команды пакета Matlab

Команда	Описание
chol	Факторизация Холецкого
det	Вычисление определителя матрицы
eig	Вычисление собственных значений матрицы
fft	Быстрое преобразование Фурье
inv	Вычисление обратной матрицы
lu	LU-разложение матрицы
ode45	Решение задачи Коши для системы обыкновенных дифференциальных уравнений методом Рунге–Кутты 4–5-го порядка
qr	QR разложение матрицы
transpose	Транспонирование матрицы

Таблица 9.3. Команды, обеспечивающие взаимодействие Maple и MATLAB

Команда	Описание
closelink	Закрытие сеанса MATLAB
defined	Проверка существования переменных в сеансе MATLAB
dimensions	Вычисление размеров матрицы с использованием команды Maple
evalM	Выполнение любого выражения MATLAB в его сеансе
getvar	Передача значения переменной или массива из сеанса MATLAB в сеанс Maple
openlink	Открытие сеанса MATLAB
setvar	Передача значения переменной или массива из сеанса Maple в сеанс MATLAB
size	Вычисление размеров матрицы с использованием команды MATLAB
square	Проверка квадратности матрицы

Проиллюстрируем взаимодействие двух вычислительных сред и использование некоторых перечисленных команд.

Определим случайную числовую матрицу при помощи команды пакета linalg:

```
> A:=linalg[randmatrix](35,35):
```

Теперь с помощью команды пакета Matlab вычислим ее собственные числа:

```
> ev:=eig(A):
```

Полученные собственные значения из переменной ev передадим в сеанс MATLAB, определив в нем для этого одноименную переменную. Переменная MATLAB указывается в кавычках в качестве первого параметра команды setvar, а в качестве второго ее параметра выступает переменная сеанса Maple:

```
> setvar("ev",ev);
```

Теперь выполним команду MATLAB построения графика, указав ее в двойных кавычках в качестве параметра команды evalM:

```
> evalM("plot(ev,'ko')");
```

В результате появляется графическое окно MATLAB с изображением собственных значений (см. рис. 9.5). Теперь в среде MATLAB определим массив x:

```
> evalM("x=1:0.1:10");
```

Следующие две команды проверяют существование переменных x и A в сеансе MATLAB:

```
> defined("x"); defined("A");
```

true

false

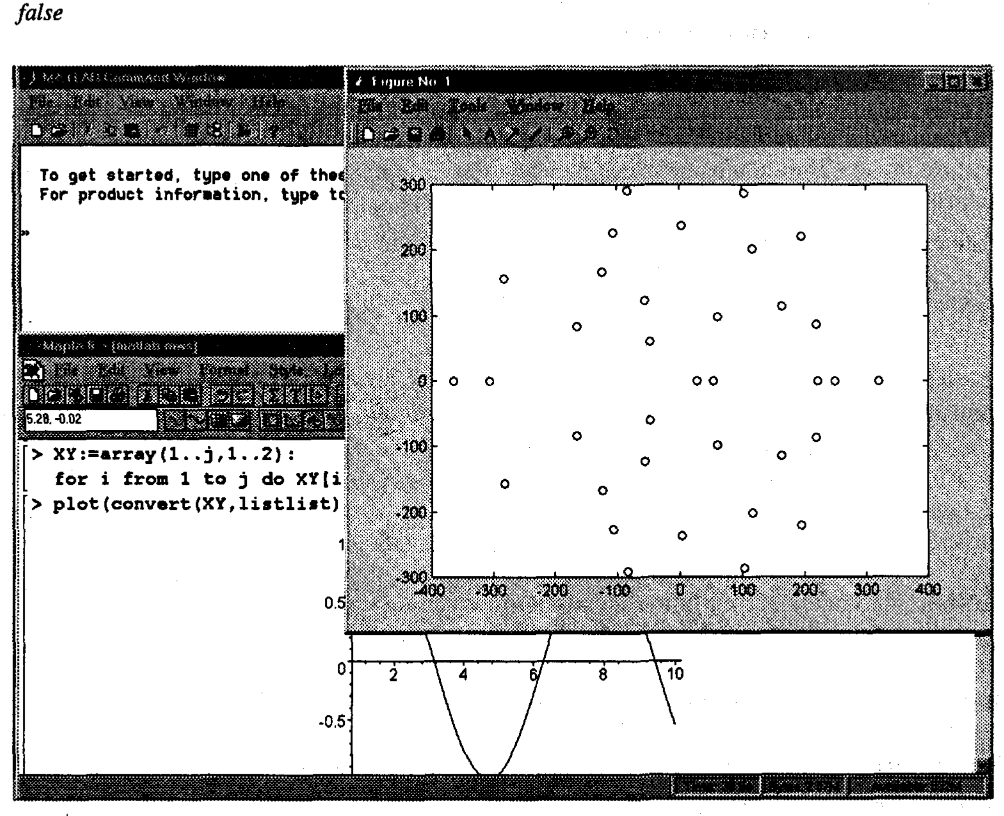


Рис. 9.5. Окна при вызове команд MATLAB из Maple. Наверху слева изображен фрагмент командного окна MATLAB, наверху справа — графическое окно MATLAB, а внизу — окно Maple

Теперь определим переменную Maple x и передадим ей содержимое массива x из сеанса MATLAB. Имя переменных MATLAB должно быть указано в двойных кавычках в качестве параметра команды getvar:

```
> x:=getvar("x");
```

$$x := \begin{bmatrix} \text{91 Element Row Vector} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran_order} \end{bmatrix}$$

Следующая команда выведет размерность массива из x сеанса MATLAB:

```
> size("x"); j:=%[2]:
```

[1,91]

Определим двумерный массив XY размерности j×2 и присвоим каждому [i,1]-му элементу значение x[i], а [i,2]-му — значение функции sin в этой точке. Затем построим график табулированной функции средствами Maple (см. рис. 9.5):

```
> XY:=array(1..j,1..2):
for i from 1 to j do XY[i,1]:=x[i]; XY[i,2]:=sin(x[i]) od:
> plot(convert(XY,listlist),color=black);
```

Последняя команда примера закрывает сеанс MATLAB:

```
> closelink();
```

Приведенный пример показывает, что взаимодействие двух вычислительных сред реализовано достаточно просто и эффективно, что значительно повышает эффективность их использования в математическом исследовании. Еще один пример совместного применения Maple и MATLAB для решения реальной задачи математического моделирования дан в главе 10 «Примеры решения задач».

Работа с Maple из среды Excel

Если на компьютере установлена программа Microsoft Excel 2000, то к ней можно подключить программу Maple 6. Такое подключение происходит автоматически при использовании англоязычной версии Windows, а для иных версий подключение происходит после открытия файла wmimplex.xla (находится в подкаталоге Maple Excel) в Excel. После подключения становится возможным вызов из Excel любой команды Maple и в меню появится дополнительная кнопочная панель Maple (см. рис. 9.6). В табл. 9.4 описаны все значки этой панели. Для активизации некоторых команд необходимо подключить соответствующий пакет, обратившись к значку установки параметров Maple в Excel. С помощью меню, которое изображено на правой части рис. 9.6, имеется возможность подключения или отключения специализированных пакетов Maple либо пакетов, разработанных пользователем.

Таблица 9.4. Значки меню Maple в Excel

Значок	Описание
	Копировать данные из Excel в Maple
	Копировать данные из Maple в Excel
	Установка параметров использования Maple в Excel
	Подсказка по установленным в Excel функциям Maple
?	Подсказка по командам Maple и его использованию в среде Excel

Обращение к команде maple_comm Maple осуществляется в Excel командой

```
=Maple("maple_comm")
```

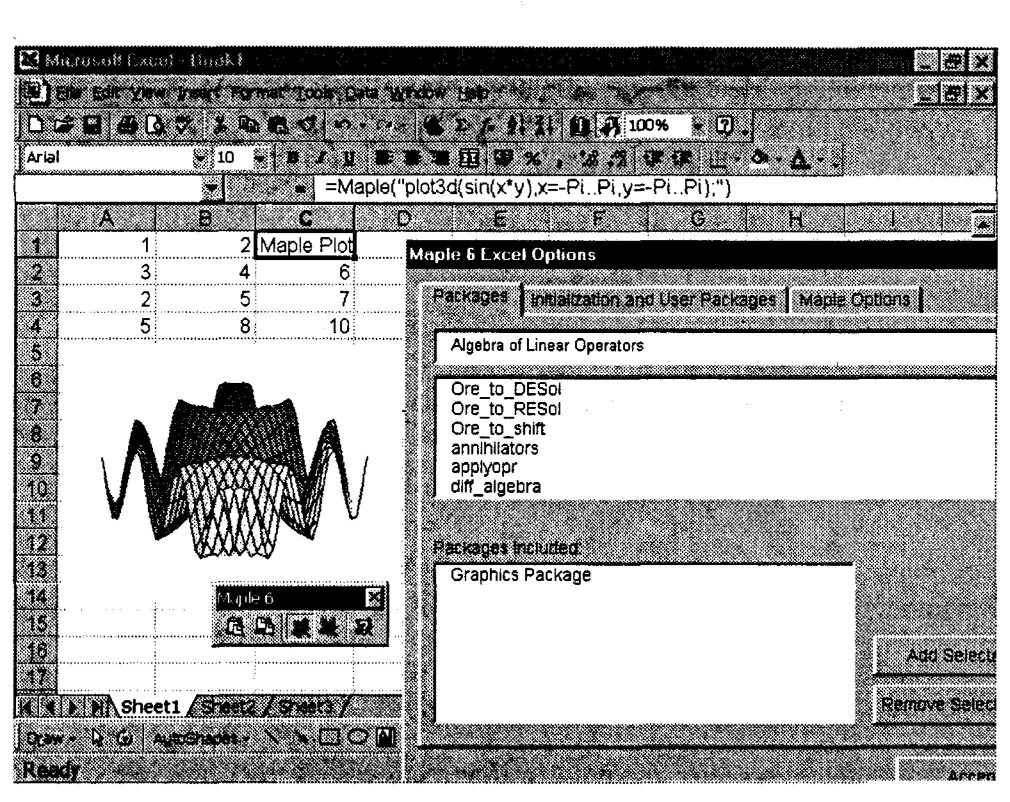


Рис. 9.6. Окно Excel с встроенным меню Maple и меню установки параметров Maple

На рис. 9.6 приведен пример обращения к команде plot3d рисования поверхности. Отметим, что сама программа Maple при этом не запускается, а подключаются только необходимые dll библиотеки.



Примеры решения задач

В данной главе рассмотрен ряд примеров исследования реальных математических задач средствами Maple. Назначение этих примеров — показать возможности Maple для решения сложных проблем и проиллюстрировать действие разнообразных команд. Для приложения Maple выбраны задачи из различных областей математики — анализа, численных методов, дифференциальных уравнений и математического моделирования.

В первом примере — задаче разложения функции одной переменной в ряд Фурье — демонстрируется создание и использование процедур. Следующий пример иллюстрирует построение численных методов при помощи средств компьютерной алгебры и демонстрирует возможности различных манипуляций с формулами. В третьем примере анализируется система обыкновенных дифференциальных уравнений, описывающая колебания регулируемого маятника с одной степенью свободы, показываются возможности изучения динамических систем и визуализации результатов с помощью Maple. Последний пример посвящен исследованию модели взаимодействия активно перемещающегося хищника и пассивной жертвы в замкнутом ареале. Здесь используется разнообразный математический аппарат, в том числе взаимодействие Maple с MATLAB и выполнение громоздких символьных вычислений в среде Maple.

Разложение функции в ряд Фурье

В этом разделе мы предлагаем процедуру, которая для заданного математического выражения выписывает отрезок ряда Фурье. Напомним, что если `f(x)` — периодическая функция от `x` с периодом `2L`, по теории рядов Фурье она может быть описана следующим выражением:

$$f(x) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}\left(a_k \cos\left(\frac{\pi k x}{L}\right) + b_k \sin\left(\frac{\pi k x}{L}\right)\right)$$

при условии, что на f(x) наложены довольно слабые ограничения. Величины a_k и b_k определяются по формулам:

$$a_k = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{\pi k x}{L}\right)dx$$

$$b_k = \frac{1}{L}\int_{-L}^{L} f(x)\sin\left(\frac{\pi k x}{L}\right)dx$$

Ряды Фурье описаны во многих книгах, от учебников по высшей математике и математическому анализу (см., например, [58]) до специальных монографий [74].

Опишем процедуру с именем fourieseries:

```
> fourieseries:=proc(f::algebraic,x,L,n::integer,
    cofs::evaln) local i,k,a,b,s;
```

Входными параметрами процедуры являются: f – алгебраическое выражение (тип algebraic), для которого нужно выписать ряд Фурье, x – переменная, по которой проводится разложение. Отрезок по x определяется входным параметром L, то есть x изменяется на интервале [-L,L], а целое число n задает количество пар членов ряда, где под парой понимается сумма синусов и косинусов. Нулевой член разложения в процедуре вычисляется всегда. Перечисленные параметры являются обязательными, а параметр cofs может отсутствовать. Если он указан, то в результате работы процедуры будет сформирована переменная типа list с именем cofs, содержащая записи вида: [базисная функция, соответствующий коэффициент]. Для вычислений внутри процедуры используются пять локальных переменных i, k, a, b, s.

Соответствие типов фактических параметров типам, описанным в заголовке процедуры, проверяется Maple при обращении к fourieseries. Проверка других условий, наложенных на входные параметры, содержится в первых строках процедуры. В частности, проверяется интервал разложения, который не должен иметь нулевой длины, положительность количества членов разложения n и то, что в качестве второго параметра указано имя переменной. При несоответствии фактических параметров наложенным на них ограничениям выводится сообщение об ошибке при помощи команды ERROR:

```
  if l=0 then
   ERROR("Интервал разложения нулевой длины!"); fi;
 if n<0 then
   ERROR("Число членов разложения должно быть >=0 !"); fi;
 if not type(x,"symbol") then
   ERROR("Второй параметр не является именем переменной!");
  fi;
```

Далее следует содержательная часть процедуры. Переменной s присваивается нулевой член ряда Фурье:

```
 s:=int(f,x=-L..L)/L/2;
```

Если при обращении к процедуре указан пятый параметр, то соответствующая переменная очищается, формируется первый ее элемент (нулевой член ряда), и инициализируется счетчик элементов:

```
if nargs=5 then cofs:='cofs'; cofs[1]:=[1,s]; k:=2 end if;
```

Затем в цикле происходит вычисление последующих коэффициентов ряда Фурье, суммы из n членов ряда Фурье и пополнение переменной cofs (если задан пятый фактический параметр процедуры):

```
for i from 1 to n  do
a:=int(f*cos(i*Pi*x/L),x=-L..L)/L; .
b:=int(f*sin(i*Pi*x/L),x=-L..L)/L;
s:=s+a*cos(i*Pi*x/L)+b*sin(i*Pi*x/L);
if nargs=5 then k:=k+1; cofs[k]:=[cos(i*Pi*x/L),a];
               k:=k+1; cofs[k]:=[sin(i*Pi*x/L),b];
          end if;  od;
```

Последние строки формируют результат. Сначала переменная cofs преобразуется к типу list, а затем в качестве результата работы процедуры возвращается значение локальной переменной s, которая содержит отрезок ряда Фурье:

```
if nargs=5 then cofs:=convert(cofs,list); fi;
 s; end;
```

После ввода описанной процедуры, Maple выводит ее текст (представлен в уменьшенном виде):

```
fourieseries := proc (f::algebraic, x, L, n::integer, cofs::evaln)
local i, k, a, b, s;
    if L = 0 then ERROR (`• •• ••••• ••••• •••• ••••••• ••••• !`) end if;
    if n < 0 then ERROR (`• •••• ••••• ••••• ••••••• •• •• • >=0 !`) end if;
    if not type(x, symbol) then ERROR (`• •• ••••• ••••••• •• ••••• •• •• •••• ••••••••• !`) end if;
    s := 1 / 2*int(f, x= -L ..L) / L;
    if nargs= 5 then cofs := 'cofs'; cofs[1] := [1, s]; k := 2 end if;
    for i to n do
        a := int(f*cos(i*π*x/ L), x= -L ..L) / L;
        b := int(f*sin(i*π*x/ L), x= -L ..L) / L;
        s := s+ a*cos(i*π*x/ L) + b*sin(i*π*x/ L);
        if nargs= 5 then k := k+ 1; cofs[k] := [cos(i*π*x/ L), a]; k := k+ 1; cofs[k] := [sin(i*π*x/ L), b] end if
    end do;
    if nargs= 5 then cofs := convert(cofs, list) end if;
    s
end proc
```

Проверим работу процедуры на нескольких примерах. Сначала обратимся к fourieseries, указав неправильные фактические параметры:

```
> fourieseries(x*y^2,y*y,1,1);
Error, (in fourieseries) Второй параметр не является именем переменной!

> fourieseries({exp(x)},y,1,1);
Error, fourieseries expects its 1st argument, f, to be of type algebraic, but received {exp(x)}
```

Определим выражение с параметром a, для которого будет строиться разложение:

```
> f:=(x-a)^2*sin(x);
```

$$f := (x-a)^2 \sin(x)$$

Выпишем первые пять членов ряда Фурье, а затем построим графики исходной функции и полученного разложения при значении параметра a=0 (см. рис. 10.1):

```
> g:=simplify(fourieseries(f,x,Pi,2));
```

$$g := -2\,a + a\cos(x) + \frac{1}{3}\sin(x)\,\pi^2 - \frac{1}{2}\sin(x) + a^2\sin(x) + \frac{4}{3}\,a\cos(2\,x) - \frac{16}{9}\sin(2\,x)$$

```
> a:=0:
> fig1:=plot(g,x=-Pi..Pi,color=black, style=point,
         symbolsize=12, symbol=circle, numpoints=30,
         legend="Отрезок ряда Фурье."):
> fig2:=plot(f,x=-Pi..Pi,color=black,
         legend="Исходная функция."):
> plots[display](fig1,fig2,axes=boxed);
```

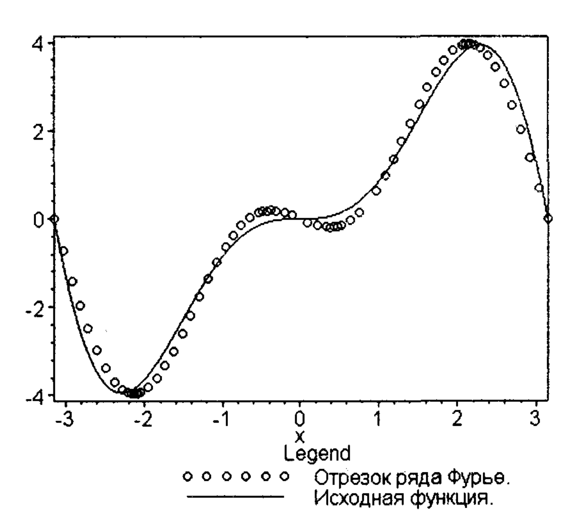


Рис. 10.1. График исходной функции и ее разложения в ряд Фурье

Обратим внимание, что при обращении к fourieseries последний необязательный параметр опущен. Из рисунка видно, что даже небольшой отрезок ряда Фурье неплохо приближает исходную функцию.

Процедура fourieseries может работать со всеми объектами Maple типа algebraic, в частности с кусочно-непрерывными функциями. Определим функцию, которая принимает значение -x при x>0, и x при x?0:

```
> f:=piecewise(x>0,-x,x);
```

$$f := \begin{cases} -x & 0 < x \\ x & \mathit{otherwise} \end{cases}$$

Обратимся к процедуре fourieseries и получим первые 11 членов ряда Фурье (нулевой член и пять пар синусов и косинусов). В качестве необязательного пятого параметра укажем переменную cc, в которой будут содержаться базисные функции и коэффициенты при них:

```
> g:=fourieseries(f,x,Pi,5,cc):
```

Теперь определим число элементов переменной cc и выведем на экран предпоследний из них:

```
> nops(cc); cc[%-1];
```

11

$$\left[\cos(5\,x),\frac{4}{25}\frac{1}{\pi}\right]$$

Закончим пример построением графиков исходной кусочно-непрерывной функции f и ее приближения отрезком ряда Фурье — рис. 10.2:

```
>fig1:=plot(g,x=-Pi..Pi,color=black,style=point,
          symbolsize=12,symbol=circle,numpoints=10,
          legend="Отрезок ряда Фурье."):
>fig2:=plot(f,x=-Pi..Pi,color=black,
          legend="Исходная функция."):
plots[display]({fig1,fig2},axes=boxed);
```

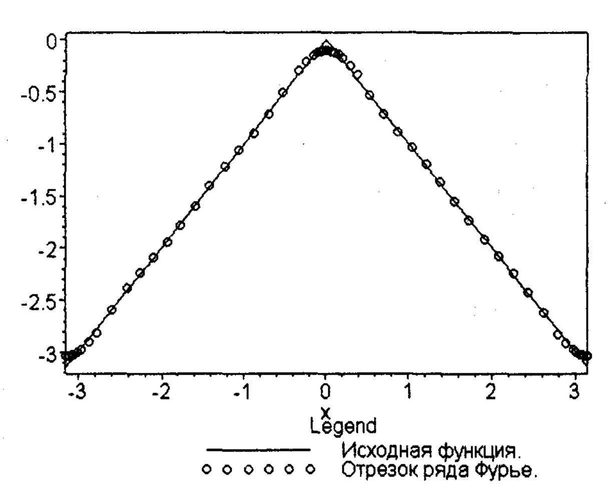


Рис. 10.2. Графики функции и ее разложения в ряд Фурье

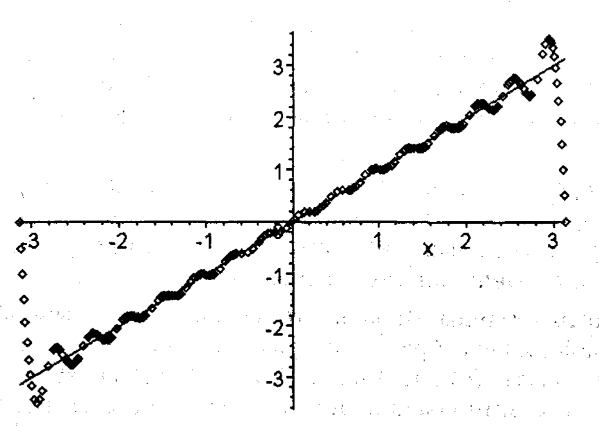


Рис. 10.3. Иллюстрация эффекта Гиббса

Приближение отрезком ряда Фурье дает хорошие результаты для периодической функции, что показали два предыдущих примера. В случае непериодической функции дело обстоит гораздо хуже, в частности может иметь место так называемый эффект Гиббса (см. [74]). Эффект состоит в том, что значение аппроксимации на некоторых интервалах всегда превосходит значение исходной функции и вблизи точек разрыва имеют место колебания аппроксимации и рост погрешности. Это иллюстрируется следующим примером разложения линейной функции (см. рис. 10.3):

```
> g:=fourieseries(x,x,Pi,15,cc):
> fig1:=plot(g,x=-Pi..Pi, symbol=diamond, symbolsize=12,
style=point,color=black):fig2:=plot(f,x=-Pi..Pi,color=black):
plots[display]({fig1,fig2});
```

Вывод формул явного метода Рунге–Кутты

Пакет Maple можно использовать как для непосредственного решения задач, так и в качестве инструмента построения и анализа численных методов. Проиллюстрируем возможности Maple для получения и решения громоздких алгебраических уравнений, возникающих при выводе формул Рунге–Кутты методом неопределенных коэффициентов.

Методы Рунге–Кутты применяются для решения дифференциальных уравнений с начальными данными (задачи Коши). Рассмотрим автономное векторное уравнение$y'=f(y)$, где штрих означает дифференцирование по времени, а y является вектор-функцией времени t. Общая формула явного метода Рунге–Кутты с n этапами (стадиями) имеет следующий вид [47,71]:

$$k_1 = hf(y_0),\quad k_j = hf(y_0 + \sum_{l=1}^{n-1} a_{jl} k_l),\quad j = 2 \div n,$$

$$y_1 = y_0 + \sum_{j=1}^{n} b_j k_j$$

Здесь h — шаг интегрирования, а параметры b^j, a^{ij} подбираются так, чтобы обеспечить наиболее высокий из возможных порядок метода. Порядок или точность метода определяется по наименьшей степени шага h в выражении невязки — разности между точным и приближенным решением за один шаг. В настоящее время развита соответствующая теория и известны многочисленные реализации методов, см. ссылки в [71].

Разберем непосредственный вывод исходя из общей схемы метода и ограничимся получением формулы для двухстадийного метода. Отметим, что в программе для краткости отсутствует какая-либо оптимизация при получении и последующем решении системы уравнений для искомых коэффициентов.

Будем приводить команды Maple в математической и обычной нотациях. Напомним, что в Maple для преобразования уже набранных строк ввода к нужной нотации достаточно нажать (отжать) значок X на панели Context Bar. В математической нотации оператор суммирования записывается в виде суммы, а индексы обретают естественный вид.

В начале программы полезно поставить команду restart, чтобы очистить все назначения. Определим переменную n для указания числа этапов (стадий):

> *restart*; $n := 2$

Подготовим формулу для вычисления решения в точке t+h:

> $App := y + \left(\sum_{j=1}^{n} 'b_j\, k_j'\right)$

$$App := y + b_1\, k_1 + b_2\, k_2$$

Для сравнения покажем, как данная команда выглядит в обычной нотации:

```
> App:=y+sum("b[j]*k[j]",j=1..n):
```

Вспомогательные функции для вычисления стадий поместим в специальную переменную kk (последовательность выражений):

> $kk := \mathrm{seq}\left(k_j = h\, \mathrm{f}\left(y + \left(\sum_{i=1}^{j-1} 'a_{j,\,i}\, k_i'\right)\right), j = [\mathrm{seq}(n - m, m = 0\,..\,n - 1)]\right)$

$$kk := k_2 = h\, \mathrm{f}(y + a_{2,\,1}\, k_1),\ k_1 = h\, \mathrm{f}(y)$$

При n=2 число неизвестных коэффициентов равно трем:

> $var := \mathrm{seq}(b_j, j = 1\,..\,n), \mathrm{seq}(\mathrm{seq}(a_{j,\,i}, i = 1\,..\,j - 1), j = 1\,..\,n)$

$$var := b_1, b_2, a_{2,\,1}$$

Запишем исходное дифференциальное уравнение:

> $de := \frac{\partial}{\partial x}\, \mathrm{y}(x) = \mathrm{f}(\mathrm{y}(x))$

Сформируем последовательность выражений для производных высоких порядков функции y(x), полученных в предположении, что y(x) является решением данного дифференциального уравнения. Для этого в цикле дифференцируем уравнение и используем соотношения, накапливаемые в переменной des, для преобразования производных в правой части:

```
> des:=de:
for j to n do
  de:=diff(lhs(de),x)=subs(des,diff(rhs(de),x)):
  des:=des,de:
od:
```

Вывод сформированной последовательности производных des опустим. Например, последний элемент переменной des дает выражение третьей производной функции y(x) через производные правой части уравнения de:

```
> des[-1];
```

$$\frac{\partial^3}{\partial x^3}\, \mathrm{y}(x) = (\mathrm{D}^{(2)})(f)(\mathrm{y}(x))\, \mathrm{f}(\mathrm{y}(x))^2 + \mathrm{D}(f)(\mathrm{y}(x))^2\, \mathrm{f}(\mathrm{y}(x))$$

Для точного решения найдем разложение в ряд по h:

> $\boldsymbol{Exact} := \mathbf{convert}(\mathbf{series}(\mathbf{y}(x + h), h, n + 2), \boldsymbol{diff})$

$$Exact := y(x) + \left(\frac{\partial}{\partial x} y(x)\right) h + \frac{1}{2}\left(\frac{\partial^2}{\partial x^2} y(x)\right) h^2 + \frac{1}{6}\left(\frac{\partial^3}{\partial x^3} y(x)\right) h^3 + O(h^4)$$

Используя подготовленную переменную des, произведем замену производных:

$> E0 := \mathrm{subs}(y(x) = y, \mathrm{subs}(\{des\}, Exact))$

$E0 :=$

$$y + f(y)\,h + \frac{1}{2} D(f)(y)\,f(y)\,h^2 + \left(\frac{1}{6}(D^{(2)})(f)(y)\,f(y)^2 + \frac{1}{6} D(f)(y)^2 f(y)\right) h^3 + O(h^4)$$

Теперь разложим в ряд по h приближенное решение в точке t+h:

$> A0 := \mathrm{series}(\mathrm{subs}(kk, App), h, n+2)$

$$A0 := y + (b_1 f(y) + b_2 f(y))\,h + b_2 D(f)(y)\,a_{2,1} f(y)\,h^2 + \frac{1}{2} b_2 (D^{(2)})(f)(y)\,a_{2,1}{}^2 f(y)^2 h^3 + O(h^4)$$

Вычислим невязку и отбросим малые величины порядка n+2, превратив выражение в полином:

$> \psi := \mathrm{simplify}(\mathrm{convert}(A0 - E0, polynom))$

В дальнейшем удобно рассматривать производные разных порядков как некоторые скалярные величины. Для этого подготовим ряд вспомогательных замен:

```
> sub:=seq((D@@j)(f)(y)=F[j],j=0..n);
```

$$sub := f(y) = F_0, D(f)(y) = F_1, (D^{(2)})(f)(y) = F_2$$

Запишем последовательность введенных вспомогательных величин:

```
> co_F:=map(rhs,[sub])[];
```

$$co_F := F_0, F_1, F_2$$

Используя эти обозначения, перепишем выражение невязки, сгруппировав коэффициенты при степенях h и вспомогательных величинах F_0, F_1, F_2:

$> \psi0 := \mathrm{collect}(\mathrm{subs}(sub, \psi), [h, co_F], distributed)$

$\psi0 :=$

$$\left(-\frac{1}{2} + b_2 a_{2,1}\right) F_1 F_0 h^2 - \frac{1}{6} h^3 F_1{}^2 F_0 + \left(-\frac{1}{6} + \frac{1}{2} b_2 a_{2,1}{}^2\right) h^3 F_2 F_0{}^2 + (b_1 - 1 + b_2)\,h\,F_0$$

Из полученного выражения для невязки видно, что никаким подбором коэффициентов не удастся обнулить второе слагаемое. Таким образом, для двухстадийного метода максимум возможного — это метод второго порядка точности, который получится, если найдутся значения коэффициентов, при которых пропадут члены первого и второго порядка по h.

Соберем коэффициенты при различных степенях шага h и учтем, что при одной степени h могут быть различные комбинации производных (у нас это теперь выражения, составленные из элементов co_F). Выделение уравнений помогает осуществить следующий фрагмент программы, где используется команда type для определения типа выражения и превращения суммы в последовательность выражений:

```
> eq:=NULL:
for j to n do
  p:=coeff(psi0,h^j);
  if type(p,"+")
    then  for s in p do eq:=eq,op(1,s); od;
    else eq:=eq,op(1,p);
  fi;
od:
```

В результате уравнения будут собраны как элементы переменной eq. Выведем число уравнений и их вид:

> nops([*eq*]); *eq*

2

$$b_1 - 1 + b_2, -\frac{1}{2} + b_2\, a_{2,1}$$

Остается решить полученную систему уравнений относительно неизвестных коэффициентов:

> *so* := solve({ *eq* }, { *var* })

$$so := \{ a_{2,1} = \frac{1}{2}\frac{1}{b_2}, b_2 = b_2, b_1 = 1 - b_2 \}$$

Подставим найденные величины в невязку:

> *simplify*(*subs*(*so*, ψ))

$$-\frac{1}{24}\frac{h^3 \mathrm{f}(y)\,(-3\,(\mathrm{D}^{(2)})(f)(y)\,\mathrm{f}(y) + 4\,(\mathrm{D}^{(2)})(f)(y)\,\mathrm{f}(y)\,b_2 + 4\,\mathrm{D}(f)(y)^2\,b_2)}{b_2}$$

Итак, невязка имеет третий порядок по h. Результирующие формулы получим подстановкой. Вспомогательные функции после подстановки коэффициентов примут вид (пустые скобки позволяют превратить список в последовательность выражений):

> subs(*so*, [*kk*])$_{[\,]}$

$$k_2 = h\ \mathrm{f}\left(y + \frac{\frac{1}{2}k_1}{b_2} \right),\ k_1 = h\ \mathrm{f}(y)$$

Таким образом, для вычисления значения функции y(t) за шаг имеем однопараметрическое семейство формул:

> subs(*so*, *App*)

$$y + (-b_2 + 1)\,k_1 + b_2\,k_2$$

Итак, получено семейство формул Рунге–Кутты второго порядка с одним свободным параметром.

Если рассмотреть формулы Рунге–Кутты с четырьмя стадиями (n=4), то для метода четвертого порядка точности (выкладки опускаем) получается система семи

уравнений с десятью неизвестными параметрами. Известно множество вариантов данного метода, в частности классический метод Рунге–Кутты и так называемый метод «три восьмых» [71]. Анализируя полученную систему уравнений, можно указать еще одно решение следующего вида:

$$a_{2,1}=\frac{1}{3},\ b_2=\frac{3}{8},\ b_3=\frac{3}{8},\ b_4=\frac{1}{8},\ b_1=\frac{1}{8},\ a_{4,3}=1,\ a_{3,2}=1,\ a_{4,1}=1,\ a_{3,1}=\frac{-1}{3},\ a_{4,2}=-1$$

Подбор параметра для интегрирования Гамильтоновых систем

Для однопараметрического двухстадийного метода можно распорядиться свободным параметром так, чтобы выполнялось какое-нибудь дополнительное условие. Рассмотрим для примера случай Гамильтоновой системы, для которой движения происходят на интегральной поверхности полной энергии системы. Эта поверхность, или энергетический уровень, специфицируется заданием начальных значений координат и импульсов (точки в фазовом пространстве). Известно, что использование методов Рунге–Кутты приводит к тому, что в процессе вычислений полная энергия может меняться.

Для демонстрации эффекта различного выбора свободного параметра на поддержание постоянства энергии рассмотрим простую систему второго порядка — ангармонический маятник. Запишем гамильтониан системы:

> $restart;\ Ham := y \to \frac{y_2^{\,2}}{2}+\frac{y_1^{\,2}}{2}+\frac{y_1^{\,4}}{4}$

Теперь получим систему дифференциальных уравнений:

> $f := \mathrm{unapply}\left(\left[\frac{\partial}{\partial y_2}\mathrm{Ham}(y),\ -\left(\frac{\partial}{\partial y_1}\mathrm{Ham}(y)\right)\right], y\right)$

$f := y \to [y_2,\ -y_1 - y_1^{\,3}]$

Напишем процедуру для реализации одного шага по формуле метода Рунге–Кутты со свободным параметром b2:

```
> rk2step := proc (b2, h, y)
  local k1, k2;
      if b2 = 0 then ERROR('b2 = 0') end if ;
      k1 := h×f(y);
      k2 := h×f(y + k1/(2×b2));

    y + (1 − b2)×k1 + b2×k2
  end proc
```

Аргументами процедуры являются параметр b2, величина шага h и вектор неизвестных y (переменная типа list). Результатом является переменная типа list. Чтобы получить решение на некотором отрезке и нарисовать фазовую траекторию, подготовим процедуру вычисления m шагов:

rk2 := **proc** (θ, *h*, *y*, *m*)
local *k*, *yy*, *sol*;
  if θ = 0 **then** ERROR('θ = 0') **end if** ;
  yy := *y*;
  sol := *yy*;

  for *k* **to** *m* **do** *yy* := rk2step(θ, *h*, *yy*); *sol* := *sol*, *yy* **end do** ;
  [*sol*]
end proc

Для рисования графиков изменения гамильтониана от времени введем функцию, готовящую список из пар «время/гамильтониан»:

> $tseries := y \to [\,\mathrm{seq}([j\,h, \mathrm{Ham}(y_k)], j = 1\,..\,\mathrm{nops}(y))\,]$

Выберем начальные данные, число шагов, величину шага h и несколько значений параметра b2:

> $y := [0, .4];\ m := 250;\ h := .1;\ par := [.2, .3, .5, .85]$

Проведем расчеты при заданных переменной par значениях параметра:

> for j to nops(par) do $tH \| j$:= tseries(rk2(par_j, h, y, m)) end do

Теперь нарисуем графики изменения гамильтониана от времени для различных значений параметра b2 (см. рис. 10.4):

> plot([*tH* ‖ (1 .. nops(*par*))], 0 .. *m h*, *legend* = map(*convert*, *par*, *string*),
  labels = ["", "Ham"], *thickness* = [seq(*j*, *j* = 1 .. nops(*par*))], *color* = *black*,
  axes = *boxed*)

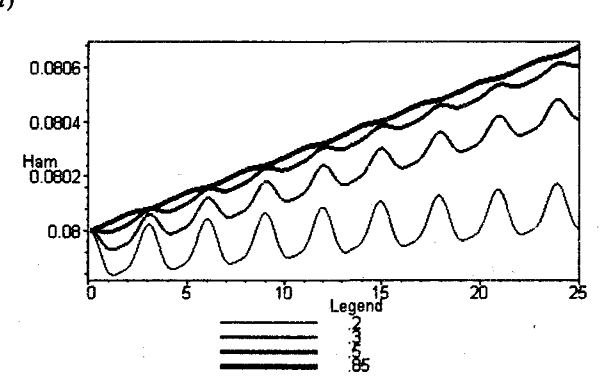


Рис. 10.4. Динамика численного решения при различных константах метода

Проведем расчеты при тех же значениях параметра b2, выбрав начальные значения, отвечающие большей величине энергии (см. рис. 10.5). Приведем соответствующие команды в обычной нотации:

```
> y:=[0.1]: m:=250: h:=.1: par:=[.2,.3,.5,0.85]:
> for j to nops(par) do tH||j:=tseries(rk2(par[j],h,y,m)):od:
> plot([tH||(1..nops(par))],0..m*h,
legend=map(convert,par,string),labels=["","Ham"],
thickness=[seq(j,j=1..nops(par))],color=black,axes=boxed);
```

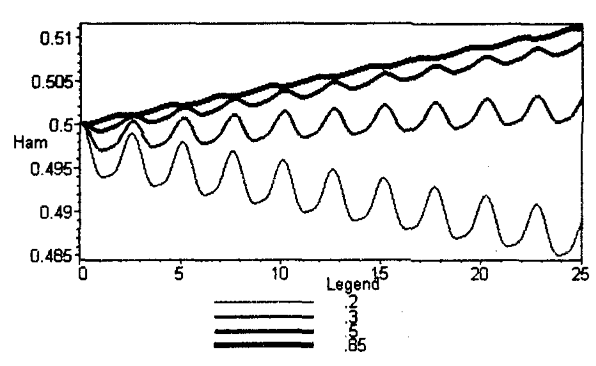


Рис. 10.5. Динамика численного решения при различных константах метода

На полученных рисунках даны кривые изменения полной энергии для различных значений параметра b2. Видно, что в зависимости от параметра метода и энергетического уровня в расчетах получается как рост, так и убывание энергии. Для выбранного энергетического уровня и заданного шага интегрирования можно подобрать значение параметра так, чтобы в расчетах происходили осцилляции гамильтониана около его начального значения.

Движение шарика в потенциальной яме

Продемонстрируем возможности пакета Maple для исследования обыкновенных дифференциальных уравнений и визуализации результатов на примере системы уравнений, описывающих движения шарика единичной массы. Сначала мы рассмотрим идеальный случай, когда трение отсутствует, а движение определяется потенциальной энергией V. Затем будет изучено влияние диссипации и внешней силы, а в заключение — действие обратной связи. Мы остановимся на ключевых моментах, таких как построение фазового портрета, анализ равновесий и их бифуркаций и др. Интересно отметить, что знаменитая система Лоренца [77] может быть записана в виде уравнения маятника с обратной связью [75].

Консервативная система

Вначале рассмотрим случай консервативной системы. Запишем выражения для потенциальной энергии V и гамильтониана H. Переменная q1 является координатой шарика, а p1 — его скоростью:

```
> V:= q1^4/4+(a-1)*q1^2/2;
```

$$V := \frac{1}{4} q1^4 + \frac{1}{2}(a-1)\, q1^2$$

```
> H:=p1^2/2+V;
```

$$H := \frac{1}{2} p1^2 + \frac{1}{4} q1^4 + \frac{1}{2}(a-1)\, q1^2$$

Для вывода системы обыкновенных дифференциальных уравнений, определяемых гамильтанианом H, воспользуемся командой из пакета Detools:

```
> with(DEtools):
> Heq:=hamilton_eqs(H);
```

$$Heq := \left[\frac{\partial}{\partial t}\,\mathrm{p1}(t) = -\mathrm{q1}(t)^3 - (a-1)\,\mathrm{q1}(t), \frac{\partial}{\partial t}\,\mathrm{q1}(t) = \mathrm{p1}(t)\right], [\mathrm{p1}(t), \mathrm{q1}(t)]$$

В результате получена система двух автономных обыкновенных дифференциальных уравнений первого порядка относительно неизвестных p1(t) и q1(t), зависящая от параметра a.

Изучим влияние параметра a на динамику шарика. Выберем два значения параметра a (a=0, a=2) и построим для них графики потенциальной энергии и фазовые портреты. Для построения фазовых портретов воспользуемся командой DEplot пакета **DEtools**. Параметры команды задают интервал интегрирования (t=-14..14), точки траекторий, шаг интегрирования (stepsize=.1), количество узлов для изображения векторного поля (dirgrid=[20,15]), а также размер стрелок, оси координат, цвет и толщину линий. Обратимся дважды к этой команде, указав различные значения параметра a и точки траекторий при t=0. Результат присвоим переменным picphase1 (a=0) и picphase2 (a=2):

```
> picphase1 := DEplot(subs(a=0,Heq[1]),Heq[2],
 t=-14..14, [ [q1(0)=1.5,p1(0)=0],[q1(0)=1,p1(0)=0],
 [q1(0)=0,p1(0)=0.02], [q1(0)=0.4,p1(0)=0],
 [q1(0)=0.8,p1(0)=0], [q1(0)=0,p1(0)=0.3],
 [q1(0)=0.6,p1(0)=0] ], stepsize=.1,dirgrid=[20,15],
arrows=small,scene=sort(Heq[2]),linecolor=black,color=gray,
 axes=none,thickness=0):
> picphase2 := DEplot(subs(a=2,Heq[1]),Heq[2],t=-14..14,
 [[q1(0)=1.5,p1(0)=0],[q1(0)=1.2,p1(0)=0],
 [q1(0)=0.8,p1(0)=0.0],[q1(0)=0.5,p1(0)=0],
 [q1(0)=0.2,p1(0)=0]],stepsize=.1,dirgrid=[20,15],
 arrows=small,
 scene=sort(Heq[2]),linecolor=black,color=gray,
 axes=none,thickness=0):
```

Теперь построим для выбранных значений параметра графики потенциальной энергии, обратившись дважды к команде plot, и присвоим результаты переменным plotH1 и plotH2:

```
> plotH1:=plot(subs(a=0,V),q1=-1.5..1.5,
  axes=none, color=black,title="a=0",
  titlefont=[COURIER,OBLIQUE,20],labels=["",""]):
> plotH2:=plot(subs(a=2,V),q1=-1.5..1.5,
  axes=none, color=black,title="a=2",
  titlefont=[COURIER,OBLIQUE,20],labels=["",""]):
```

Для совмещения картин на одном рисунке сначала объединим их в один графический объект, запретив наложение рисунков параметром insequence=true, и присвоим результат переменной tempplot:

```
> tempplot:=plots[display](plotH1,plotH2,picphase1,picphase2,
                                         insequence=true):
```

Следующая команда выводит все четыре картины на одном рисунке (см. рис. 10.6):

```
> plots[display](tempplot,insequence=false);
```

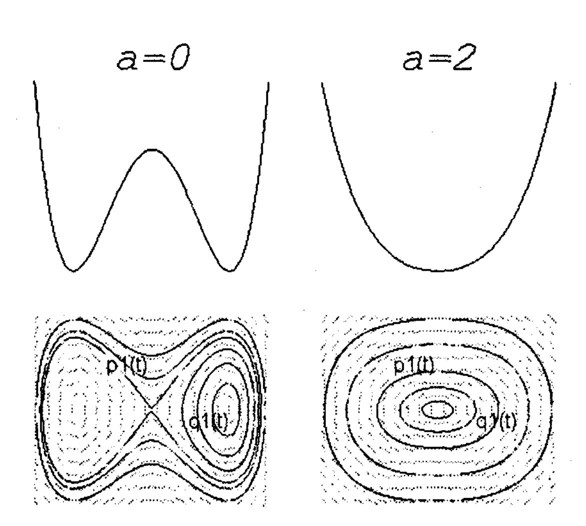


Рис. 10.6. Графики потенциальной энергии и соответствующие фазовые портреты для двух значений параметров

Из полученных фазовых портретов видно, что колебания для консервативной системы не затухают. Для a<1 существуют два типа движений: одни происходят в окрестности равновесий, соответствующих локальным минимумам потенциальной энергии, а другие охватывают оба эти равновесия. При a>1 все колебания происходят вокруг единственного равновесия, соответствующего минимуму потенциальной энергии.

Система с диссипацией

Перейдем к исследованию системы с трением. В уравнение для импульса добавим диссипативный член, который в зависимости от знака параметра b определяет либо диссипацию, либо накачку энергии. Переменной Deq присвоим измененную систему обыкновенных дифференциальных уравнений:

```
> Deq:=[Heq]:
  Deq[1][1]:=lhs(Deq[1][1])=rhs(Deq[1][1])-b*p1(t):
  Deq:=Deq[1],Deq[2];
```

$$Deq := \left[\frac{\partial}{\partial t}\,\mathrm{p1}(t) = -\mathrm{q1}(t)^3 - (a-1)\,\mathrm{q1}(t) - b\,\mathrm{p1}(t), \frac{\partial}{\partial t}\,\mathrm{q1}(t) = \mathrm{p1}(t)\right], [\mathrm{p1}(t), \mathrm{q1}(t)]$$

Теперь изучим влияние трения на динамику системы. Для этого построим фазовые портреты для двух значений параметра a (0 и 2) и b=0.1 и присвоим результаты переменным pic1 и pic2:

```
> pic1:=DEplot(subs(a=0,b=0.1,Deq[1]),Deq[2],t=0..80,
[[q1(0)=1.5,p1(0)=0.0],[q1(0)=0.1,p1(0)=0.0]],stepsize=0.1,
 dirgrid=[20,15],arrows=small, scene=sort(Heq[2]),
 color=gray,axes=none,thickness=0, title="a=0 b=0.1",
 linecolor=black,titlefont=[COURIER,BOLD,15]):
> pic2:=DEplot(subs(a=2,b=0.1,Deq[1]),Deq[2],t=0..80,
[[q1(0)=1.5,p1(0)=0.0]],stepsize=.1,dirgrid=[20,15],
 arrows=small,
```

```
scene=sort(Heq[2]),linecolor=black,color=gray,
 axes=none,thickness=0,title="a=2 b=0.1",
 titlefont=[COURIER,BOLD,15]):
```

Следующие команды выводят на одном рисунке (см. рис. 10.7) оба фазовых портрета:

```
> with(plots,display):
> display(display([pic1,pic2],insequence=true),
                              insequence=false);
```

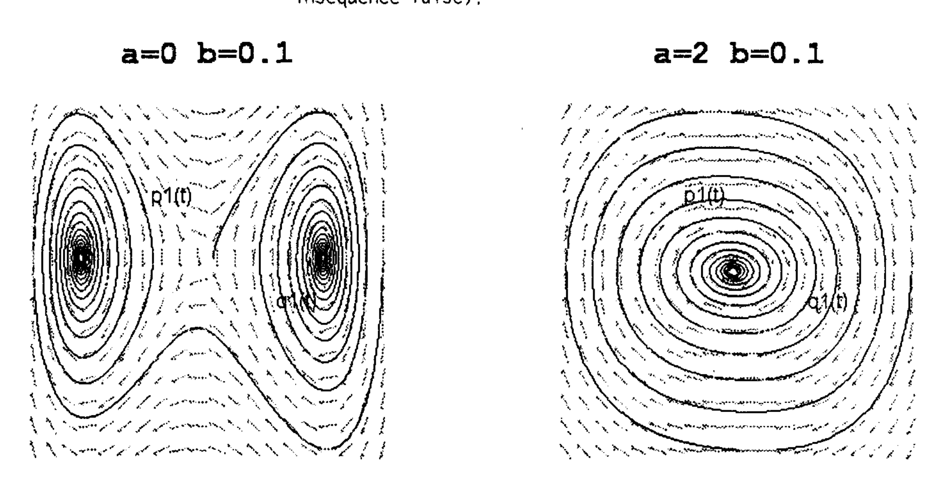


Рис. 10.7. Фазовые портреты для двух наборов параметров

Вычисления показывают, что для каждого равновесия существует множество начальных данных, из которых решение к нему стремится. При введении трения равновесия системы становятся асимптотически устойчивыми и исследование их устойчивости можно провести средствами Maple.

Равновесия и их устойчивость

Остановимся на изучении равновесий и их устойчивости для диссипативной системы. Вначале получим уравнения равновесий и найдем их координаты:

```
> Equil_eq:=expand(subs(q1(t)=q1,p1(t)=p1,Deq[1]));
```

$$Equil_eq := [0 = -q1^3 - q1\ a + q1 - b\ p1, 0 = p1]$$

```
> Equil:=solve(convert(Equil_eq,set),{p1,q1});
```

$$Equil := \{p1 = 0, q1 = 0\}, \{p1 = 0, q1 = \mathrm{RootOf}(_Z^2 + a - 1, label = _L1)\}$$

Далее вычислим матрицу линеаризации системы и характеристический многочлен:

```
> A:=linalg[jacobian](map(rhs,Equil_eq),[p1,q1]);
```

$$A := \begin{bmatrix} -b & -3\,q1^2 - a + 1 \\ 1 & 0 \end{bmatrix}$$

```
> chp:=linalg[charpoly](A,lambda);
```

$$chp := \lambda^2 + \lambda\, b + 3\, q1^2 + a - 1$$

Решая полученное уравнение, найдем собственные значения и присвоим их переменной eig_eq:

```
> eig_eq:=solve(chp,lambda);
```

$$eig_eq := -\frac{1}{2}b+\frac{1}{2}\sqrt{b^2-12\,q1^2-4\,a+4},\ -\frac{1}{2}b-\frac{1}{2}\sqrt{b^2-12\,q1^2-4\,a+4}$$

Теперь, подставив значения параметров в полученные выражения, легко вычислить координаты любого равновесия и изучить его устойчивость. Отметим, что для всех равновесий координата p1 равна нулю и в выражениях для собственных чисел фигурирует только q1. Выделим из множества равновесий координаты q1, а затем с помощью команды map2 только их значения:

```
> Equil_q1:={Equil[1][2],allvalues(Equil[2][2])};
```

$$Equil_q1 := \{\, q1 = 0,\ q1 = -\sqrt{-a+1},\ q1 = \sqrt{-a+1}\,\}$$

```
> Equil_plot:=map2(op,2,Equil_q1);
```

$$Equil_plot := \{\, 0,\ \sqrt{-a+1},\ -\sqrt{-a+1}\,\}$$

Представим зависимость координаты q1 равновесия и характера его устойчивости от параметров a и b графически. Опишем процедуру Equil_vis, которая для данного равновесия строит поверхность q1 на интервале aI[-1.4,2.6] и bI[-1,1]. Входным параметром процедуры является номер равновесия. Внутри процедуры используется локальная переменная f1, которой присваивается максимальное собственное значение для данного равновесия. Для вывода поверхности воспользуемся командой plot3d пакета Maple, а для задания цвета введем функцию ff. Функция ff принимает значение 0.4, если максимальное собственное значение положительно, и 0.9 в противном случае. Таким образом, на рисунке устойчивые равновесия будут обозначены более светлым цветом:

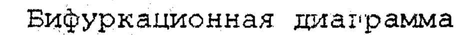
```
> Equil_vis:=proc(i::integer) local f1,ff;
   f1:=max(evalc(Re(subs(p1=0,q1=Equil_plot[i],eig_eq[1]))),
   evalc(Re(subs(p1=0,q1=Equil_plot[i],eig_eq[1]))));
   ff:=piecewise(evalf(f1)>0,0.4,0.9);
   plot3d(Equil_plot[i],a=-1.4..2.6,b=-1..1,
              color=[ff,ff,ff],grid=[31,40]);
 end;
```

Equil_vis := **proc** (*i*::*integer*)
local *f1*, *ff*;
$f1 := \max(\mathrm{evalc}(\Re(\mathrm{subs}(p1 = 0, q1 = Equil_plot[i], eig_eq[1]))),$
$\mathrm{evalc}(\Re(\mathrm{subs}(p1 = 0, q1 = Equil_plot[i], eig_eq[1]))));$
$ff := \mathrm{piecewise}(0 < \mathrm{evalf}(f1), .4, .9);$
$\mathrm{plot3d}(Equil_plot[i], a = -1.4 .. 2.6, b = -1 .. 1, color = [ff, ff, ff],$
$grid = [31, 40])$
end proc

Обратившись трижды к описанной процедуре, построим бифуркационную диаграмму для трех равновесий и характерных интервалов изменения параметров (см. рис. 10.8):

```
> display(seq(Equil_vis(i),i=1..3),axes=framed,
```

```
title="Бифуркационная диаграмма",orientation=[50,50],
titlefont=[COURIER,14],labels=["a","b","q1"]);
```



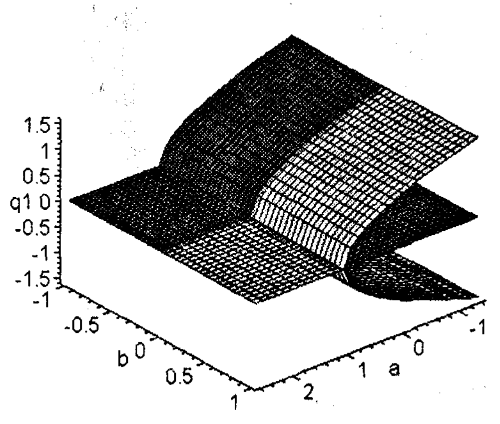


Рис. 10.8. Поверхности равновесий в пространстве параметров и переменной q1

Видно, что нулевое равновесие существует при всех значениях параметров и устойчиво при a>1 и b>0, а ненулевые равновесия существуют только при a<1 и устойчивы, когда параметр b отрицателен. При a=1 происходит бифуркация, в результате которой нулевое равновесие обменивается устойчивостью с двумя ненулевыми.

Внешнее воздействие

Исследуем влияние периодического возмущения на поведение диссипативной системы. Для этого в уравнение для импульса добавим синусоидальное внешнее воздействие:

```
> Veq:=[Deq]:
  Veq[1][1]:=lhs(Veq[1][1])=rhs(Veq[1][1])+c*sin(w*t):
  Veq:=Veq[1],Veq[2];
```

$$Veq := \left[\frac{\partial}{\partial t}\mathrm{p1}(t) = -\mathrm{q1}(t)^3 - (a-1)\,\mathrm{q1}(t) - b\,\mathrm{p1}(t) + c\sin(w\,t),\ \frac{\partial}{\partial t}\mathrm{q1}(t) = \mathrm{p1}(t)\right],$$
$$[\mathrm{p1}(t), \mathrm{q1}(t)]$$

В зависимости от значений амплитуды c и частоты w в системе могут наблюдаться различные периодические режимы, а в некоторых случаях реализуется хаотическое движение шарика. Следующая команда демонстрирует хаотический режим (см. рис. 10.9), наблюдающийся при a=0, b=0.1, c=0.3, w=0.5:

```
> DEplot3d(subs(a=0,b=0.1,c=0.3,w=0.5,Veq[1]),Veq[2],
      t=0..280,[[q1(0)=1.5,p1(0)=0.0]],stepsize=0.1,
      scene=[t,Heq[2][1],Heq[2][2]],linecolor=black,
      color=gray,axes=none,thickness=0,axes=boxed,
      orientation=[-170,70]);
```

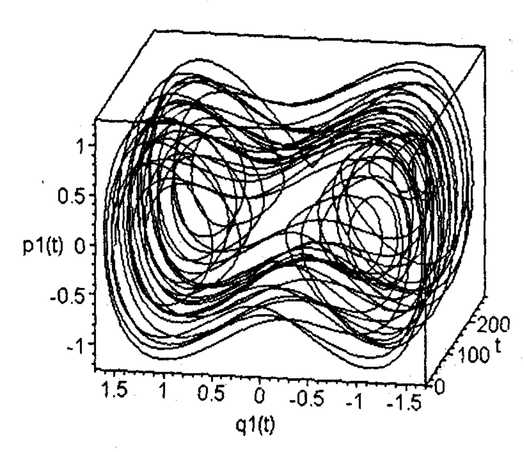


Рис. 10.9. Неавтономный хаотический режим

Система с обратной связью

Последней частью исследования является анализ системы с обратной связью. Здесь параметр a становится переменной, удовлетворяющей дифференциальному уравнению. Система уравнений имеет вид:

```
> Ceq:=convert(subs(a=a(t),Deq[1]),set) union
  {diff(a(t),t)=-epsilon*(delta*a(t)-q1(t)^2)},
  convert(Deq[2],set) union {a(t)}:
```

$$Ceq := \{ \frac{\partial}{\partial t} \mathrm{p1}(t) = -\mathrm{q1}(t)^3 - (\mathrm{a}(t) - 1)\,\mathrm{q1}(t) - b\,\mathrm{p1}(t),\ \frac{\partial}{\partial t} \mathrm{a}(t) = -\varepsilon\,(\delta\,\mathrm{a}(t) - \mathrm{q1}(t)^2),$$
$$\frac{\partial}{\partial t} \mathrm{q1}(t) = \mathrm{p1}(t) \},\ \{ \mathrm{a}(t), \mathrm{q1}(t), \mathrm{p1}(t) \}$$

Полученная система трех обыкновенных дифференциальных уравнений описывает движение шарика в меняющейся во времени потенциальной яме. Напомним, что форма потенциальной ямы зависит от переменной a. Представим решение в удобном для физической трактовки виде с помощью Maple.

Сначала используем анимационные возможности Maple для визуализации возможных режимов движения шарика. Выберем значения параметров, при которых в системе реализуется хаотический аттрактор, и решим соответствующую задачу Коши. Для вычисления мы используем метод Рунге–Кутты 7–8-го порядка с выводом результата для значений времени заданных массивом val_t:

```
> val_t:=array(1..400):
 for i from 1 to 400 do val_t[i]:=(i-1)*0.25: od:
numsol:=dsolve(subs(epsilon=1,delta=0.1,b=0.4,Ceq[1]) union
   {q1(0)=0.2,p1(0)=-0.2,a(0)=0.8},Ceq[2],
   type=numeric,value=val_t,method=dverk78):
 eval(numsol[1,1]);
```

$[t, \mathrm{q1}(t), \mathrm{a}(t), \mathrm{p1}(t)]$

Теперь проиллюстрируем движение шарика в изменяющейся потенциальной яме. Для этого напишем две процедуры: ball для изображения самого шарика и figpic для рисования графика потенциальной энергии и шарика в n-й момент времени:

```
> ball := proc(x,y) local i;
   PLOT(POLYGONS([seq([ evalf(x+0.13*cos(Pi*i/10)/2),
   evalf(0.02+y+0.04*sin(Pi*i/10)/2) ], i = 1..20)]));
 end:
 figpic:=proc(n);
   plots[display](plot(subs(a=numsol[2,1][n,3],V),
   q1=-1.0..1.0, color=black,thickness=2),
   ball(numsol[2,1][n,2],
   subs(a=numsol[2,1][n,3],q1=numsol[2,1][n,2],V)));
 end:
```

Результатом следующей команды будет мультфильм, состоящий из 400 кадров. Мы приводим на рисунке (см. рис. 10.10) только четыре из них.

```
> plots[display]([seq(figpic(i),i=1..400)],
                              insequence=true,axes=NONE);
```

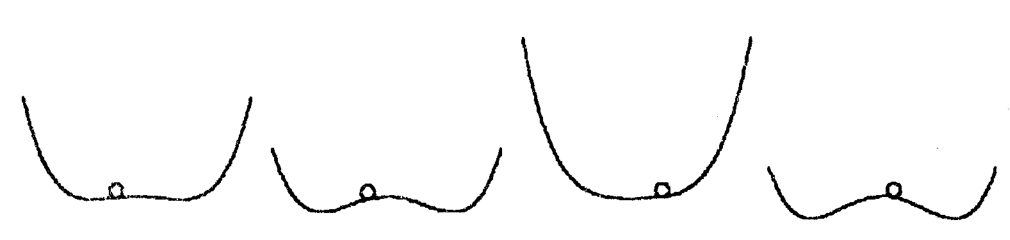


Рис. 10.10. Четыре кадра мультфильма, иллюстрирующего динамику шарика в изменяющейся потенциальной яме

В заключение представим на одном рисунке изменяющуюся во времени потенциальную энергию и траекторию шарика. Вычислим для каждого значения переменной t координаты шарика и присвоим их массивам val_V и val_q1. В качестве координат здесь фигурирует значение потенциальной энергии и переменная q1:

```
> val_V:=array(1..400): val_q1:=array(1..400):
 for i from 1 to 400 do
   val_V[i]:=evalf(subs(a=numsol[2,1][i,3],
                              q1=numsol[2,1][i,2],V));
   val_q1[i]:=numsol[2,1][i,2]:  od:
```

Создадим набор точек spcurve, приближающих траекторию шарика, и присвоим переменной pic1 результат выполнения команды spacecurve, соединяющей эти точки в трехмерном пространстве:

```
> spcurve:=[seq([(i-1)*0.25,val_q1[i],val_V[i]],i=1..400)]:
> pic1:=plots[spacecurve](spcurve,color=black,thickness=5,
                              orientation=[-125,20]):
```

Теперь сформируем набор точек, каждая из которых является значением потенциальной энергии в различные моменты времени на интервале q1=[-0.65,0.65]:

```
> i:='i': sdat:=[seq([seq([(j-1)*0.25, -0.65+i*0.065,
        subs(a=numsol[2,1][j,3], q1=-0.65+i*0.065,V)],
        i=0..20)],j=1..400)]:
```

При помощи команды surfdata построим поверхность для данных sdat, а результат присвоим переменной pic2. С помощью ambientlight и light определим параметры подсветки и назначим серый цвет поверхности при помощи shading:

```
> pic2:=plots[surfdata](sdat,style=PATCHNOGRID,
          ambientlight=[1,1,1],light=[0,0,1,0,1,1,1],
          shading=ZGREYSCALE,orientation=[-125,20]):
```

Наконец, при помощи команды display получим рисунок (см. рис. 10.11), на котором в различные моменты времени изображены форма потенциальной ямы и положение шарика:

```
> display({pic1,pic2},axes=framed, titlefont=[HELVETICA,12]
   labels=["t","q1","V"],labelfont=[TIMES,BOLD,14]);
```

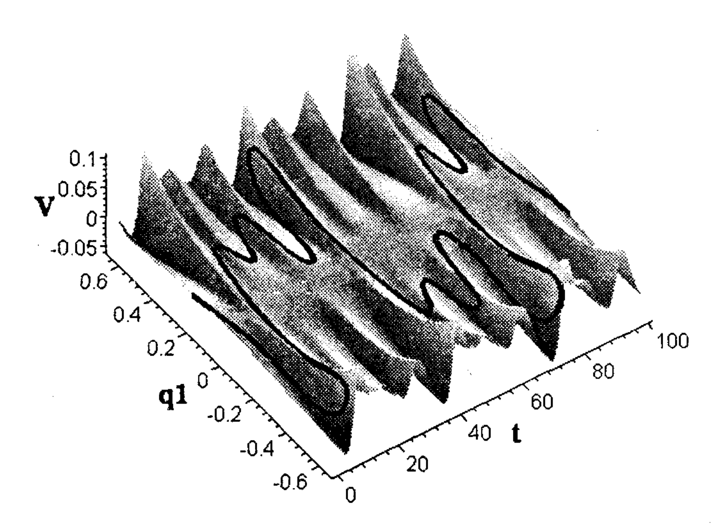


Рис. 10.11. Изменение во времени формы потенциальной энергии и траектории шарика

Исследование уравнений в частных производных методом Галеркина

В предыдущем разделе математическая модель движения материальной точки выражалась системой обыкновенных дифференциальных уравнений. Математические модели динамики неоднородных по пространственным переменным величин обычно записываются в виде уравнений в частных производных. Такое математическое описание используется в механике деформируемых тел, физике, гидродинамике, химии и других науках (см., например, книгу [59] и библиографию в ней). В этом разделе мы проиллюстрируем один из возможных способов исследования уравнений в частных производных с помощью Maple и MATLAB.

В некоторых случаях уравнения в частных производных поддаются аналитическому решению, но часто приходится прибегать к численному анализу. Это означает аппроксимацию исходных уравнений в частных производных системой обыкновенных дифференциальных или разностных уравнений и решение полученного конечномерного аналога. Широко распространены два способа дискретизации уравнений в частных производных: аппроксимация конечными разностями [65] и методы типа Галеркина [68]. С помощью конечных разностей приближенная система выписывается явно без особых трудностей, а получение галеркинских аппроксимаций зачастую связано с громоздкой аналитической работой, которую позволяет автоматизировать Maple.

Идея метода Галеркина состоит в том (см., например [68]), что приближенное решение разыскивается в виде отрезка ряда по известным базисным функциям с неизвестными коэффициентами при них. Базисные функции являются функциями пространственных переменных, бесконечный набор функций должен образовывать полную систему и каждая из них должна удовлетворять граничным условиям. Неизвестные коэффициенты при базисных функциях в стационарном случае будут константами, а в случае нестационарных задач — функциями, зависящими от времени. Конечномерная система относительно неизвестных получается после подстановки отрезков рядов в исходные уравнения и проецирования результата на все присутствующие в аппроксимации базисные функции. Под проецированием здесь понимается интегрирование по пространственной области полученного выражения, умноженного на базисную функцию. Обычно при использовании метода Галеркина используют ортогональный набор функций, что значительно упрощает вывод уравнений.

При выводе галеркинской аппроксимации с помощью Maple возникает ряд технических проблем. Во-первых, приходится интегрировать выражения, состоящие из большого числа слагаемых, что требует значительных компьютерных ресурсов, а в силу их ограниченности это не всегда возможно, особенно для аппроксимаций большого порядка. Другая проблема вызвана необходимостью численного решения систем уравнений большого порядка, для чего предпочтительнее использовать не Maple, а программные средства численного анализа. Для преодоления перечисленных трудностей можно предложить интегрирование каждого слагаемого отдельно, использование рабочих файлов для записи промежуточных результатов и вызов MATLAB для численного решения полученных систем обыкновенных дифференциальных уравнений.

Модель «активный хищник — жертва»

Начало математическому описанию взаимодействия популяций было положено В. Вольтерра в 1930-х годах (см. [79]), и с того времени математическое моделирование в биологии активно развивается. Спектр задач и методов исследования в этой области очень широк (см. [78]): изучаются механизмы взаимодействий, динамика популяций, влияние факторов среды и др. В качестве иллюстративного примера нами выбрана математическая модель взаимодействия двух популяций, которая была предложена и исследована в работе [46].

Исследуем модель взаимодействия двух популяций: способных направленно перемещаться хищников и не способных к этому жертв. Рассматривается вытянутый ареал обитания популяций, который можно считать отрезком и без потери общности принять его длину равной единице. Предполагается, что ускорение направленного перемещения популяции хищников пропорционально градиенту плотности популяции жертвы. Локальные процессы размножения и гибели жертв описываются классической моделью Лотки–Вольтерра, а для хищников вклад этих процессов не рассматривается. Переменными модели являются следующие функции, зависящие от времени t и пространственной координаты x: c(t,x) — отклонение плотности популяции хищников от среднего значения C=const, s(t,x) — плотность популяции жертв, v(t,x) — скорость перемещения хищников. Теперь приведем уравнения модели и присвоим их переменным в Maple.

Уравнения для скорости жертв и плотностей хищников и жертв при надлежащем масштабировании имеют вид [46]:

```
> eq1:=diff(v(t,x),t)=kappa*diff(s(t,x),x)+
                            delta||0*diff(v(t,x),x$2);
```

$$eq1 := \frac{\partial}{\partial t} v(t,x) = \kappa\left(\frac{\partial}{\partial x} s(t,x)\right) + \delta 0\left(\frac{\partial^2}{\partial x^2} v(t,x)\right)$$

```
> eq2:=diff(c(t,x),t)=-C*diff(v(t,x),x)-diff(c(t,x)*v(t,x),x)+
                            delta||1*diff(c(t,x),x$2);

> eq3:=diff(s(t,x),t)=s(t,x)*(S-s(t,x)-c(t,x))+
                            delta||2*diff(s(t,x),x$2);
```

$$eq3 := \frac{\partial}{\partial t} s(t,x) = s(t,x)\,(S - s(t,x) - c(t,x)) + \delta 2\left(\frac{\partial^2}{\partial x^2} s(t,x)\right)$$

```
> S:=1-C:
```

В этих уравнениях присутствуют следующие параметры: C — средняя плотность хищников, коэффициент κ характеризует миграционную активность, δ1, δ2, δ3 — диффузионные коэффициенты. Ареал сообщества предполагается замкнутым, то есть потоки особей через границу местообитания отсутствуют, что описывается следующими граничными условиями:

$$v|_{x=0,1} = \frac{\partial c}{\partial x}\Big|_{x=0,1} = \frac{\partial s}{\partial x}\Big|_{x=0,1} = 0$$

Решение системы (eq1, eq2, eq3), будем разыскивать в виде отрезка ряда из N членов по линейно-независимым функциям, удовлетворяющим граничным условиям. Для v(t,x) такими функциями являются sin(i π x), а для c(t,x) и s(t,x) — cos(i π x), где i = 0, 1, …

Определим два набора ортонормированных базисных функций ϕ_i и ψ_i:

```
> phi:=i->sqrt(2)*sin(i*Pi*x);
```

$$\phi := i \rightarrow \sqrt{2}\,\sin(i\,\pi\,x)$$

```
> psi:=i->if i=0 then 1 else sqrt(2)*cos(i*Pi*x) fi;
```

ψ := **proc** (i)
option *operator*, *arrow*;
if $i = 0$ **then** 1 **else** sqrt(2)×cos(i×π×x) **end if**
end proc

Решения будем разыскивать в виде:

```
> v(t,x):=Sum(v[i](t)*phi(i),i=1..N);
```

$$v(t, x) := \sum_{i=1}^{N} v_i(t) \sqrt{2} \sin(x \pi i)$$

```
> c(t,x):=Sum(c[i](t)*psi(i),i=1..N);
```

$$c(t, x) := \sum_{i=1}^{N} c_i(t) \sqrt{2} \cos(x \pi i)$$

```
> s(t,x):=s0(t)+Sum(s[i](t)*psi(i),i=1..N);
```

$$s(t, x) := s0(t) + \left(\sum_{i=1}^{N} s_i(t) \sqrt{2} \cos(x \pi i) \right)$$

Меняя N, можно изучить динамику для различных аппроксимаций и определить необходимый для сходимости результатов порядок приближения. Мы представим вывод уравнений для 12 членов ряда, для чего зададим параметр N:

```
> N:=12:
```

Очевидно, что суммарное количество неизвестных коэффициентов vi(t), ci(t), si(t) будет следующим:

```
> Nn:=3*N+1:
```

Для хранения дифференциальных уравнений введем массив

```
> equat:=array(1..Nn):
```

Вывод галеркинской системы

Перед выводом галеркинской системы обсудим процедуру получения этой системы в Maple. Проецирование на базисные функции, как уже отмечалось, сводится к интегрированию выражений, состоящих из суммы произведений базисных функций с символьными коэффициентами. Интегрирование громоздких выражений может быстро исчерпать всю оперативную память, что повлечет использование виртуальной памяти и приведет практически к «зависанию» компьютера. Рассмотрим пример интегрирования суммы произведений тригонометрических функций с символьными коэффициентами. Переменной test присвоим сумму 14 слагаемых и оценим оперативную память, необходимую для интегрирования этого выражения:

```
> test:=sum(a[i]*sin(i*Pi*x)*cos(i*Pi*x)*sin((i+1)*Pi*x)*
                                   sin(2*i*Pi*x),i=1..14):
> Mbyte:=evalf(kernelopts(bytesalloc)/1024/1024):
> int(test,x=0..1): expand(%);
```

$$\frac{32}{165}\frac{a_2}{\pi}+\frac{\frac{128}{1155}a_4}{\pi}+\frac{\frac{288}{3689}a_6}{\pi}+\frac{\frac{512}{8487}a_8}{\pi}+\frac{\frac{800}{16269}a_{10}}{\pi}+\frac{\frac{1152}{27755}a_{12}}{\pi}+\frac{\frac{1568}{43665}a_{14}}{\pi}$$

```
> evalf(kernelopts(bytesalloc)/1024/1024)-Mbyte;
```

2.062122345

Итак, для интегрирования суммы 14 слагаемых понадобилось около 2 Мбайт оперативной памяти. Если попытаться так вычислять сумму 30 подобных слагаемых, то это приведет к «зависанию» компьютера с 32 Мбайт ОЗУ. С этим можно бороться, интегрируя не все выражение, а каждое его слагаемое отдельно:

```
> Mbyte:=evalf(kernelopts(bytesalloc)/1024/1024):
  nn:=nops(test): res:=0:
  for i from 1 to nn do
    tmp:='tmp': tmp:=int(op(i,test),x=0..1):
    res:=res+tmp: od:
```

В результате применения этих команд оперативной памяти потребовалось почти в семь раз меньше, чем при интегрировании сразу всего выражения:

```
> evalf(kernelopts(bytesalloc)/1024/1024)-Mbyte;
```

.312442780

Такие же проблемы с использованием оперативной памяти возникают и при выводе галеркинской системы большой размерности. Поэтому ниже непосредственное интегрирование громоздких выражений заменено интегрированием каждого слагаемого и их суммированием. Однако при большом N и это не позволит провести все необходимые выкладки за один сеанс работы с Maple в силу ограниченности оперативной памяти компьютера. Поэтому предусмотрим проверку достаточности оперативной памяти и сохранение полученных уравнений в файле при ее нехватке. При повторном запуске программы эта информация будет считана и вывод уравнений продолжится.

Опишем процедуру CheckMemory, которая проверяет количество использованной за сеанс памяти (в мегабайтах), записывает массив с уравнениями equat и счетчик полученных уравнений NEQ в файлы в случае превышения величины Nmbyte и осуществляет выход из сеанса Maple:

```
> CheckMemory:=proc() local ibyte;
   ibyte:=evalf(kernelopts(bytesalloc)/1024/1024);
       if ibyte>NMbyte then
                save equat,"c:\\temp\\eqn.txt";
                save NEQ,"c:\\temp\\neq.txt";
                quit;   fi;    end:
```

При первом обращении к программе файл **eqn.txt** должен отсутствовать, а файл **neq.txt** должен содержать запись:

```
NEQ:=0;
```

Следующие две команды считывают уравнения и значение счетчика из соответствующих файлов. Мы приводим результаты работы этих команд при первом обращении к программе:

```
> read("c:\\temp\\eqn.txt"):
Error, unable to read "c:\temp\eqn.txt"
```

Сообщение об ошибке здесь не должно пугать, так как при первом обращении еще нет файла с уравнениями

```
> read("c:\\temp\\neq.txt"):
```

$NEQ := 0$

Теперь перейдем непосредственно к выводу галеркинской системы обыкновенных дифференциальных уравнений. В первую очередь раскроем все суммы в уравнениях:

```
> eq1:=value(eq1):  eq2:=value(eq2):  eq3:=value(eq3):
```

Следующий цикл вычисляет уравнения для коэффициентов $v_i(t)$. Для этого переменную eq1 проецируем последовательно на каждую функцию ϕ_i. Напомним, что в нашем случае проекция определена следующим образом:

$$\int_0^1 \mathrm{eq1}\,\phi_i\,dx$$

При выводе уравнений интегрируем отдельно левую и правую части eq1:

```
> for i from 1 to N do
   if i>NEQ then
        equat[i]:=int(lhs(eq1)*phi(i),x=0..1)
              =int(rhs(eq1)*phi(i),x=0..1);
    NEQ:=NEQ+1; fi; print(equat[i]); od:
```

В результате получим N уравнений относительно коэффициентов $v_i(t)$. Приведем только первое из них:

$$\frac{\partial}{\partial t} v_1(t) = -\pi^2\,\delta 0\,v_1(t) - \pi\,\kappa\,s_1(t)$$

После подстановки отрезков рядов во второе уравнение eq2 непосредственное проецирование его на базисные функции в силу громоздкости выражения приведет к трудностям, описанным выше. Поэтому преобразуем правую часть уравнения eq2 в сумму слагаемых, воспользовавшись командой collect, и определим число слагаемых в левой и правой частях уравнения:

```
> i:='i':
> eeq2:=collect(rhs(value(eq2)),[seq(c[i](t),i=1..N),
              seq(v[i](t),i=1..N)], distributed):
      i2:=nops(eeq2); i1:=nops(lhs(eq2));
```

$i2 := 168$

$i1 := 12$

Итак, в результате преобразований правая часть состоит из 168 слагаемых, а левая — из 12. Теперь при выводе дифференциальных уравнений относительно коэффициентов $c_i(t)$ будем отдельно интегрировать произведение каждого слагаемого на базисную функцию. С помощью процедуры CheckMemory проверим количество использованной оперативной памяти после вывода каждого уравнения. В процедуре CheckMemory используется глобальная переменная NMbyte, которая задает мак-

симальный допустимый объем использованной оперативной памяти. Присвоим ей значение 12 Мбайт:

```
> NMbyte:=12:
> for i from 1 to N do
          if (i+N)>NEQ then
       lhu:='lhu'; rhu:='rhu'; lhu:=0;
       for j from 1 to i1 do
                        lhu:=lhu+
         int(op(j,lhs(eq2))*psi(i),x=0..1) od;
                  rhu:=0;
                  for j from 1 to i2 do
                        rhu:=rhu+
          int(op(j,eeq2)*psi(i),x=0..1) od;
       equat[i+N]:=lhu=rhu; NEQ:=NEQ+1; CheckMemory();
    fi;
    print(lhs(equat[i+N]));
  od;
```

В результате выполнения этих циклов будут получены следующие N галеркинских уравнений. Для примера представим левую часть последнего уравнения:

$$\frac{\partial}{\partial t} c_{12}(t)$$

Аналогичным образом проведем выкладки для уравнения eq3, предварительно разбив его на слагаемые. Отметим, что это уравнение оказывается еще более громоздким и при N=12 после подстановки приближенного решения его правая часть состоит из 247 слагаемых:

```
> eeq3:=collect(rhs(eq3),[seq(c[i](t),i=1..N),
             seq(s[i](t),i=1..N)],distributed):
  i2:=nops(eeq3); i1:=nops(lhs(eq3));
```

$i2 := 247$

$i1 := 13$

```
> for i from 0 to N do
          if (i+2*N+1)>NEQ then
                  lhu:='lhu'; rhu:='rhu';  lhu:=0;
         for j from 1 to i1 do
                          lhu:=lhu+
           int(op(j,lhs(eq3))*psi(i),x=0..1) od;
       rhu:=0;
                  for j from 1 to i2 do
          rhu:=rhu+
           int(op(j,eeq3)*psi(i),x=0..1)  od;
       equat[i+2*N+1]:=lhu=rhu;
       NEQ:=NEQ+1;  CheckMemory(); fi;
    print(lhs(equat[i+2*N+1]));  od:
```

$$\frac{\partial}{\partial t} s0(t)$$

Мы привели только левую часть уравнения относительно коэффициента $s_0(t)$. В результате получено еще N+1 обыкновенных дифференциальных уравнений относительно $s_i(t)$, $i=0,1,...,N$.

Сохраним все полученные уравнения и общее их число в файлах:

```
>save equat,"c:\\temp\\eqn.txt"; save NEQ,"c:\\temp\\neq.txt";
```

Итак, построена аппроксимация исходной задачи в частных производных в виде системы 3N+1 обыкновенных дифференциальных уравнений. В примере использовано значение N=12, и нам не пришлось запускать программу несколько раз для вывода всех уравнений (потребовалось чуть больше 8 Мбайт оперативной памяти), но при N=20 это необходимо уже для компьютера с 64 Мбайт оперативной памяти.

Численное решение системы обыкновенных дифференциальных уравнений с использованием MATLAB

Следующим этапом исследования рассматриваемой задачи является численное решение полученной системы обыкновенных дифференциальных уравнений (ОДУ). Такое решение можно получить средствами Maple, но предпочтительнее, особенно для систем большого порядка, использовать программные средства численного анализа.

Рассмотрим применение MATLAB для численного интегрирования полученной системы ОДУ. В Maple проведем замену переменных, приблизив их имена к синтаксису MATLAB. Вместо переменных vi(t), ci(t), si(t) введем переменные y(k), для чего оформим множество (переменная podst), задающее нужную подстановку:

```
> podst:={seq(v[i](t)=y(i),i=1..N),
      seq(c[i](t)=y(i+N),i=1..N),s0(t)=y(2*N+1),
      seq(s[i](t)=y(i+2*N+1),i=1..N)};
```

Для численного решения значения всех параметров задачи должны быть определены. Выберем один набор величин и сформируем множество parsubs для подстановки значений параметров в уравнения:

```
> parsubs:={delta||0=0.00001,
         delta||1=0.2,delta||2=0.005,kappa=6,C=0.65};
```

$$parsubs := \{\kappa = 6, C = .65, \delta 0 = .00001, \delta 1 = .2, \delta 2 = .005\}$$

Для преобразованных уравнений опишем массив и присвоим его элементам правые части дифференциальных уравнений, в которых произведены описанные подстановки:

```
> eqbuf:=array(1..NEQ):
> for i from 1 to NEQ do  eqbuf[i]:=
     rhs(evalf(subs(parsubs,subs(podst,equat[i])))): od:
```

Для обращения к команде решения системы ОДУ пакета MATLAB необходимо сформировать файл, задающий правые части системы (см. главу 15 «Чис-

ленный анализ в MATLAB» второй части книги). Следующие команды создают процедуру на языке MATLAB (m-файл) и записывают ее в файл рабочего каталога MATLAB:

```
> open("c:\\MatlabR11\\work\\rmaple.m",WRITE):
  writeline("c:\\MatlabR11\\work\\rmaple.m",
             "function f = rmaple(t,y)"):
  writeline("c:\\MatlabR11\\work\\rmaple.m","f=[ ..."):
  for i from 1 to NEQ do
          nn:=nops(eqbuf[i]): j:=1: uu:=0:
          while j<=nn do  uu:=uu+op(j,eqbuf[i]):
      if (trunc(j/3)*3)=j
         then if j<nn then strend:="+..."
          else if i=NEQ then strend:="];"
                       else strend:="; ..." fi fi:
              writeline("c:\\MatlabR11\\work\\rmaple.m",
              cat(convert(uu,string),strend)):
      uu:=0:  fi;             j:=j+1:     od:
         if uu<>0 then
    if i<>NEQ then writeline("c:\\MatlabR11\\work\\rmaple.m",
                       cat(convert(uu,string),";..."))
           else writeline("c:\\MatlabR11\\work\\rmaple.m",
                     cat(convert(uu,string),"];"))    fi;
      fi;  od:     close("c:\\MatlabR11\\work\\rmaple.m"):
```

В результате в файле rmaple.m будут содержаться команды MATLAB, реализующие вычисление правых частей полученной системы обыкновенных дифференциальных уравнений. Приведем начало и конец созданной процедуры:

```
function f = rmaple(t,y)
f=[ ...
-.9869604404e-4*y(1)-18.84955592*y(26);...
. . .
-2.*y(37)*y(25)-6.756115171*y(37)-.7071067810*y(31)^2+...
-.7071067810*y(18)*y(31)-.7071067810*y(17)*y(32)-
1.*y(25)*y(24)+...
-.7071067810*y(13)*y(36)-1.414213562*y(30)*y(32)-
.7071067810*y(22)*y(27)+...
-1.414213562*y(26)*y(36)-.7071067810*y(20)*y(29)-
.7071067810*y(23)*y(26)+...
-.7071067810*y(15)*y(34)-1.414213562*y(28)*y(34)-
.7071067810*y(19)*y(30)+...
-.7071067810*y(16)*y(33)-.7071067810*y(14)*y(35)-
1.414213562*y(27)*y(35)+...
-.7071067810*y(21)*y(28)-1.414213562*y(29)*y(33)];
```

Перед вызовом команд MATLAB из Maple подключим необходимую библиотеку:

```
> with(Matlab):
```

Для решения задачи Коши и вычисления одного из возможных режимов существования популяции определим начальные значения неизвестных:

```
> yinit:=array(1..NEQ):
> for i from 1 to NEQ do yinit[i]:=0.01: od:
> yinit[N+1]:=0.2: yinit[2*N+1]:=0.5: yinit[2*N+2]:=0.1:
```

Теперь два раза обратимся к команде ode45 для решения системы обыкновенных дифференциальных уравнений методом Рунге–Кутты. Первый расчет проводится для установления режима, то есть полученные в его результате значения переменных являются начальными для второго расчета. По второму решению можно судить об установившемся режиме взаимодействия популяций:

```
> ti := 0:   tf := 40.0:
> (T, Ysol) := ode45("rmaple", ti..tf, yinit, "tol"=.01):
> ti := 0:  tf := 40.0:
> (T, Ysol) := ode45("rmaple", ti..tf, yinit, "tol"=.001):
```

Следующая строка выводит количество точек численного решения:

```
> nsol:=size(Ysol)[1];
```

$nsol := 13993$

Для визуализации решения вычислим аналитически две величины — среднее по отрезку значение съеденных жертв и среднее их количество:

```
> i:='i': varplot1:=subs(podst,parsubs,
         int(value(C+c(t,x))*value(s(t,x)),x=0..1));
```

$$varplot1 := y(14)\,y(25) + y(20)\,y(31) + y(19)\,y(30) + y(13)\,y(24) + y(17)\,y(28) + y(12)\,y(23) + y(16)\,y(27) + y(15)\,y(26) + .65\,y(21) + y(18)\,y(29) + y(11)\,y(22)$$

```
> varplot2:=subs(podst,parsubs, int(value(s(t,x)),x=0..1));
```

$varplot2 := y(25)$

Сформируем двумерный массив Yplot, содержащий значения переменных varplot1 и varplot2 в различные моменты времени. Отметим, что для каждого момента времени t формируется вспомогательная переменная ppp для подстановки чисел вместо соответствующих символьных переменных:

```
> Yplot:=array(1..nsol,1..2):
> for i from 1 to nsol do
   ppp:={seq(y(j)=Ysol[i,j],j=1..NEQ)};
   Yplot[i,1]:=subs(ppp,varplot1);
   Yplot[i,2]:=subs(ppp,varplot2);  od:
```

Теперь с помощью команды Maple evalM обратимся к графической команде MATLAB для вывода графика зависимости величин varplot1 и varplot2 от времени. Для этого передадим данные в сеанс MATLAB командой setvar:

```
>  setvar("Yplot",Yplot);
  evalM("plot(Yplot)");
  evalM("title "Динамика средних величин по времени" ");
```

В результате в графическом окне MATLAB появится следующий график — рис. 10.12.

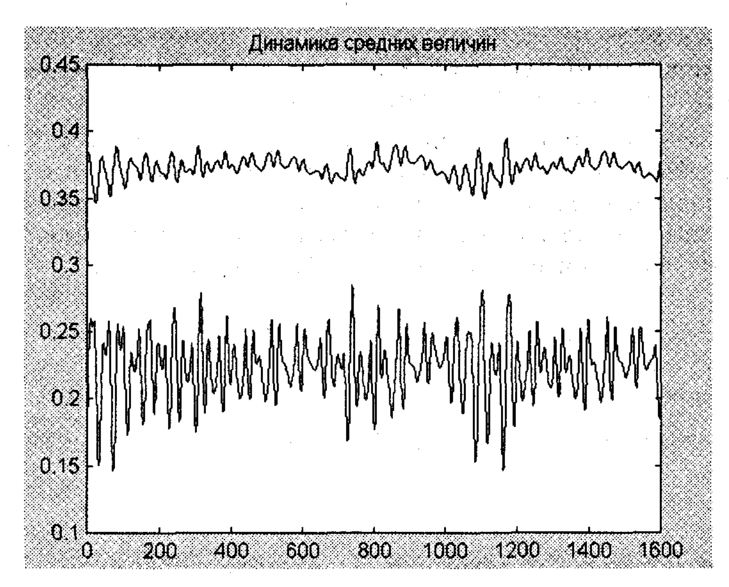


Рис. 10.12. Изменение среднего количества хищников и жертв во времени

Теперь закроем сеанс MATLAB:

```
> Matlab[closelink]();
```

Последний рисунок 10.13 подготовим средствами Maple. Для построения фазовой траектории в осях varplot1 и varplot2 воспользуемся командой plot с необходимыми опциями:

```
> plot(convert(Yplot,listlist),color=black,axes=BOXED,
  title="Хаотический режим", titlefont=[COURIER,BOLD,13]);
```

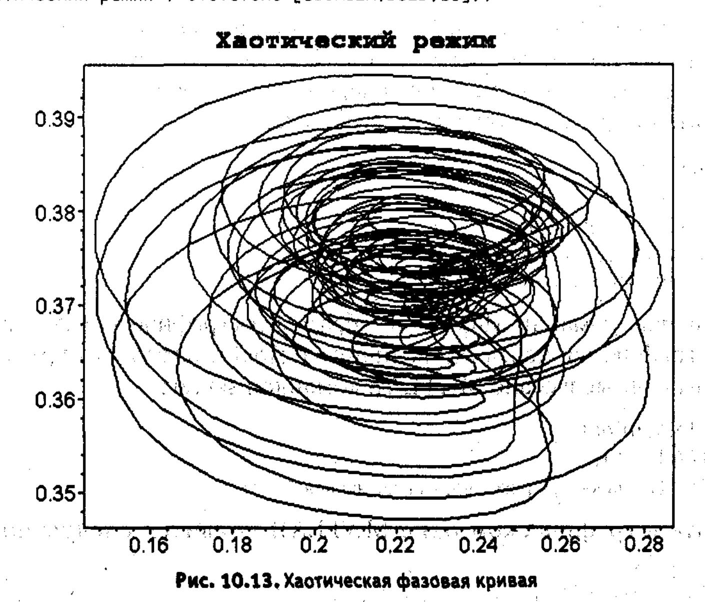


Рис. 10.13. Хаотическая фазовая кривая

Результаты вычислений демонстрируют один из возможных типов поведения системы из хищников и жертв при сделанных в математической модели предположениях. В частности, оказалось, что даже в случае простого ареала высокая миграционная активность хищника может порождать хаотизацию динамики популяционного сообщества (см. [46]).

В этом примере показано, как можно эффективно использовать описываемые в книге программные продукты для решения достаточно сложных научных проблем. Понятно, что описанная схема может быть адаптирована для применения метода Галеркина к исследованию других систем дифференциальных уравнений в частных производных.



ЧАСТЬ Расчеты в среде MATLAB

- Работа в MATLAB
- Элементы языка MATLAB
- Матричные вычисления
- Графика MATLAB
- Численный анализ в MATLAB
- Программирование в MATLAB
- Расширения MATLAB
- Дополнения и примеры

Исторически MATLAB разрабатывался как диалоговая среда для матричных вычислений (MATrix LABoratory). Со временем пакет был оснащен хорошей графической системой, дополнен средствами компьютерной алгебры от Maple и усилен библиотеками команд (или Toolboxes), предназначенными для эффективной работы со специальными классами задач. Теперь MATLAB — мощный инструмент для проведения исследований. Владея им, можно браться за сложные задачи, проводя вычисления качественно и достаточно быстро, используя мощный математический аппарат и удобную диалоговую среду MATLAB.

Сегодня MATLAB фактически является стандартным расчетным средством и инструментом для многочисленных инженерных и технических разработок. Этому способствует богатая библиотека команд и свой язык программирования, дающий пользователю возможности автоматизации вычислений, в частности через добавление новых команд (функций) — m-файлов и подключение своих программ на языке С.

В состав MATLAB входят интерпретатор команд, графическая оболочка, редактор-отладчик, профилер, библиотеки команд, компилятор, символьное ядро пакета Maple для проведения аналитических вычислений, математические библиотеки MATLAB на C/C++, Web-сервер, генератор отчетов и богатый инструментарий (Toolboxes). По-прежнему поддерживая диалоговый режим для простых вычислений, MATLAB превратился в среду программирования математических и инженерных задач, включая разработку сложных программ с развитым графическим интерфейсом.

Наше изложение ориентировано на пользователя, работающего в среде Windows для персональных компьютеров IBM. Однако большая часть материала посвящена языку и командам, не привязанным к конкретной платформе. Коротко перечислим технические требования к компьютеру и программному обеспечению, необходимые для работы с MATLAB. Для версии 5.3:

- необходимы процессоры 486, Pentium и выше, графический адаптер, поддерживающий минимум 256 цветов;
- нужны операционные системы Microsoft Windows 95, Windows 98, Windows NT 4.0 или Windows 2000;
- для работы нужно 16 Мбайт ОЗУ;
- размер дискового пространства зависит от платформы и операционной системы. При нехватке места на диске программа The MathWorks Installer проинформирует пользователя об этом;
- для работы также нужны программа Acrobat Reader для чтения документации в виде pdf-файлов и Netscape Communicator или Internet Explorer для просмотра гипертекстовой документации (HTML-файлов).

Для MATLAB версии 6.0 требования повысились, в первую очередь это касается процессора (как минимум Pentium) и ОЗУ (от 64 Мбайт), а для эффективной работы рекомендуется 128 Мбайт.

Цель данной главы состоит в том, чтобы познакомить читателя с пакетом MATLAB, описав его возможности и предоставив сведения, достаточные для самостоятельной работы с пакетом. В главе описаны интерфейсы реализации MATLAB 5.3 и новой версии — 6.0, примеры выполнены для обеих версий, а результаты представлены для версии 5.3, поскольку язык MATLAB практически не изменился.

11 ГЛАВА Работа в MATLAB

Среда MATLAB включает интерпретатор команд на языке высокого уровня, графическую систему, пакеты расширений и реализована на языке C. MATLAB постоянно модернизируется, при этом расширяются возможности системы меню, совершенствуются старые и добавляются новые команды. Меню частично дублируют ряд команд и облегчают взаимодействие с многочисленными инструментами MATLAB. По-прежнему вся работа организуется через командное окно (Command Window), которое появляется при запуске программы matlab.exe. В процессе работы данные располагаются в памяти (Workspace), создаются графические окна для изображения кривых, поверхностей и других графиков. Рассмотрим на простых примерах функционирование вычислительной среды MATLAB.

Командное окно

В командном окне в режиме диалога проводятся вычисления и активируются элементы среды MATLAB. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает введенное и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках.

Начнем с элементарных операций, чтобы проиллюстрировать интерактивный режим работы. Знаком (») в тексте будем помечать строки ввода команд MATLAB, а результат будет располагаться на последующих строках. Введем матрицу второго порядка и присвоим ее переменной A:

```
» A=[1 2; 3 4]
A =
     1     2
     3     4
```

Найдем обратную матрицу:

```
» inv(A)
ans =
```

```
  -2.0000    1.0000
   1.5000   -0.5000
```

Поскольку не было указано, к чему относится результат последней операции, то MATLAB присвоил полученную квадратную матрицу стандартной переменной ans (сокращение от Answer). Умножим обратную матрицу на квадрат матрицы A:

```
» ans*A^2
ans =
     1     2
     3     4
```

Вычислим определитель матрицы A:

```
» a=det(A)
a =
    -2
```

Окно с результатами проделанных вычислений представлено на рис. 11.1.

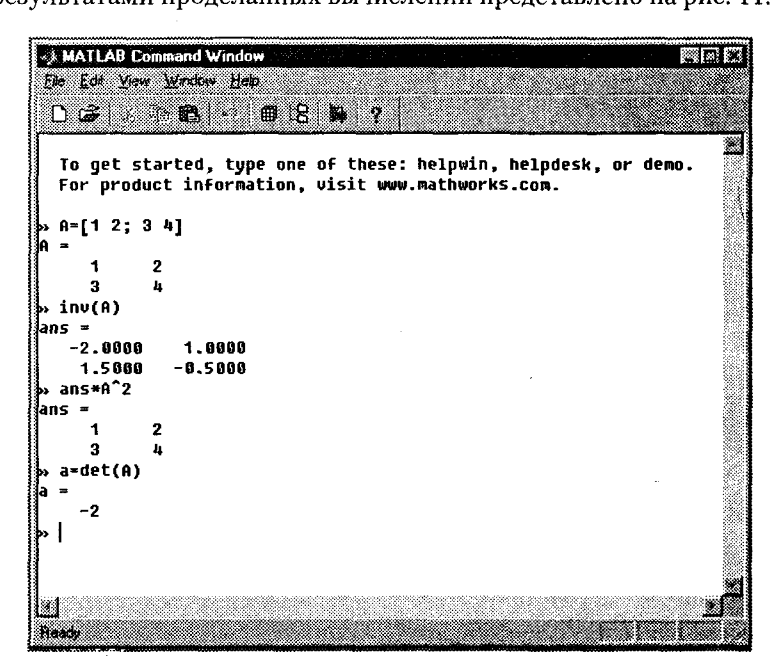


Рис. 11.1. Основное окно MATLAB

Чтобы просмотреть список используемых в текущем сеансе имен переменных, достаточно выполнить команду who. Команда whos выводит список переменных вместе с информацией о размерности, плотности заполнения и типе переменных:

```
» whos
  Name      Size           Bytes  Class
  A         2x2               32  double array
  a         1x1                8  double array
  ans       2x2               32  double array
Grand total is 9 elements using 72 bytes
```

Заметим, что результат вычисления определителя представлен в списке переменных как массив размерности 1×1. Ту же информацию получим, обратившись к пункту меню File/Show Workspace или к значку Workspace Browser. Появляется специальное окно для просмотра перечня данных рабочей области, см. рис. 11.2.

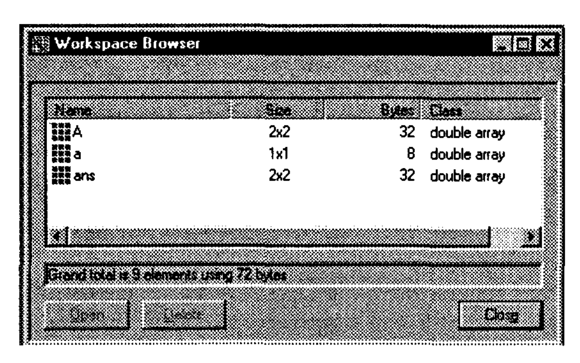


Рис. 11.2. Окно Workspace Browser

Если затем в окне Workspace Browser щелкнуть два раза на идентификаторе нужной переменной, то в MATLAB Editor/Debugger (программа medit.exe, см. рис. 11.3) появится содержимое этой переменной и станет доступным для изменения. Такое редактирование возможно для матриц (двумерных массивов), но не распространяется на многомерные массивы и структуры.

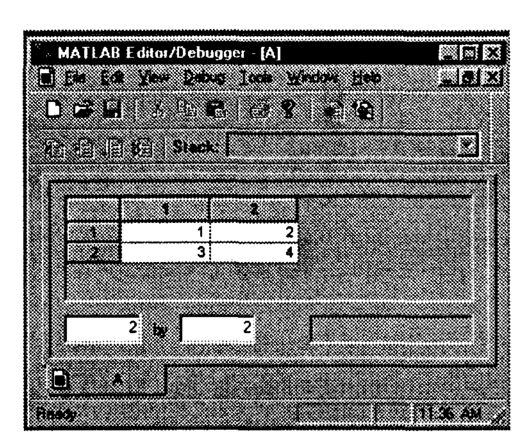


Рис. 11.3. Окно редактора Editor/Debugger

Те же вычисления можно выполнить, подготовив m-файл, называемый также файлом-сценарием или script-файлом, и запустив его на выполнение.

Наберем в любом редакторе файл first.m следующего содержания:

```
A=[1 2; 3 4]
inv(A)
ans*A^2
a=det(A)
```

и поместим его в подкаталог MATLAB \work или \bin. Теперь в строке ввода наберем слово

```
» first
```

В командном окне появится знакомые нам строки с результатами выполнения записанных в файле команд. Таким образом, в MATLAB можно приготовить файл с программой, протестировать его при помощи редактора-отладчика и выполнить, указав в строке ввода командного окна имя файла. О разработке m-файлов и удобном редакторе-отладчике medit, входящем в состав пакета, подробнее говорится в главе «Программирование в MATLAB», а пока, чтобы представить еще один элемент среды MATLAB, вернемся в командное окно и выполним команду построения графика функции exp(-0.2x)sin(x^2)

```
» fplot("exp(-0.2*x)*sin(x^2)",[0 4*pi])
```

В квадратных скобках указан интервал изменения переменной x, а функция задана как строковая переменная (в апострофах) для синтаксического анализа интерпретатором MATLAB. В результате выполнения графической команды fplot появится отдельное окно с графиком (рис. 11.4).

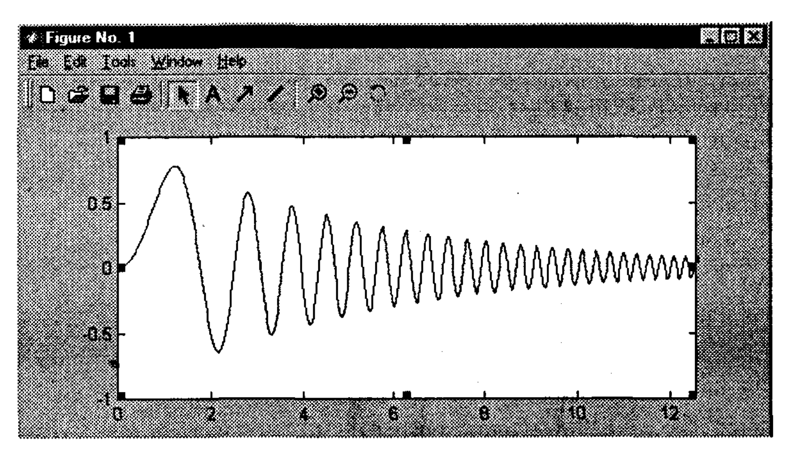


Рис. 11.4. Графическое окно

Это окно имеет свою систему меню, речь о котором пойдет в главе 14 «Графика MATLAB». Пока же рассмотрим основное (командное) окно MATLAB и его систему меню.

Система меню

Представленное на рис. 11.1 меню для MATLAB 5.3 состоит из пяти пунктов, назначение которых описано в табл. 11.1.

Система меню достаточно проста. Больше всего строк в меню File, которое состоит из стандартных для Windows (New, Open, Print) и ряда специфических пунктов, см. табл. 11.2.

Таблица 11.1. Система меню основного окна

Пункты	Назначение
File	Команды работы с файлами, свойствами окон, печать
Edit	Операции правки, команда очистки рабочей области
View	Включение/отключение строки состояния
Window	Список окон
Help	Справка, примеры и демонстрации

Таблица 11.2. Пункты меню File

Пункты	Назначение
Open Selection	Открытие файла, соответствующего выделенной функции, в окне редактора
Run script	Запуск m-файла
Load Workspace...	Считывание данных в рабочую область
Save Workspace As...	Сохранение данных рабочей области
Show Workspace	Просмотр данных в рабочей области
Show Graphics Property editor	Свойства графического редактора
Show GUI Layout Tool	Свойства графического интерфейса
Set Path	Подключение каталогов
Preferences	Назначение шрифтов, форматов и др.

Меню Edit содержит стандартные возможности (Undo, Cut, Copy, Paste, Clear, Select All) и команду очистки всех переменных в сеансе (Clear Session).

Меню View состоит из одного пункта-переключателя (Toolbar).

Меню Window позволяет переключаться от одного окна к другому, причем основное окно имеет номер 0, а нумерация графических окон начинается с 1.

Меню Help и работу справочной системы рассмотрим в следующем разделе.

Панель основного окна содержит значки часто используемых операций: создание нового и открытие существующего m-файла, операции правки, команда отмены ввода (Undo), просмотр рабочей области (Workspace Browser), подключенные каталоги (Path Browser) и справка (Help Window).

Справочная система

В справочной системе MATLAB имеется несколько способов получения информации о командах и работе с ними:

- команда help;
- меню Help;
- гипертекстовая система Help Desk.

Список разделов, объединяющих близкие по тематике команды MATLAB, можно получить, набрав в строке ввода

```
help
```

Список всех команд раздела с именем TOPIC выводится по команде

```
help TOPIC
```

Наконец, описание команды с именем COMMAND будет получено, если набрать

```
help COMMAND
```

Например, справка о команде вычисления определителя (детерминанта) квадратной матрицы выглядит следующим образом (пустые строки пропускаем):

```
» help det
DET    Determinant.
    DET(X) is the determinant of the square matrix X.
    Use COND instead of DET to test for matrix singularity.
    See also COND.
Overloaded methods
    help sym/det.m
```

Заметим, что команда help при описании функции выводит имя функции большими буквами, чтобы выделить имя из остального текста. Однако все функции пакета пишутся строчными буквами, которые и нужно использовать при написании команд. Вообще же, пакет различает большие и малые буквы, так что переменные Ab и aB различны.

Для Windows окно справки (Help Window) становится доступно через меню Help при нажатии соответствующего значка, а также при запуске в строке ввода команды

```
helpwin
```

Для того чтобы получить информацию по разделу TOPIC, следует выполнить команду

```
helpwin TOPIC
```

Приведем список разделов, сопроводив их краткими пояснениями. По этому перечню можно судить о возможностях, предоставляемых пакетом. Команды общего назначения и функции, реализующие взаимодействие пользователя со средой MATLAB, находятся в разделе general.

Все, относящееся к языку, — операторы, символы, конструкции, а также наборы команд работы со строками, временем и другим — содержится в шести разделах, которым соответствуют одноименные каталоги в директории \toolbox\matlab (см. табл. 11.3).

Следующая группа разделов представляет графику MATLAB (см. табл. 11.4). В них сосредоточены команды двумерной и трехмерной графики, презентационная графика, а также базисные команды, дающие полный доступ ко всем возможностям графической системы и позволяющие создавать для приложений графический пользовательский интерфейс (GUI — Graphic User Interface).

Математические разделы содержат стандартный набор математических функций, библиотеку специальных функций и богатую коллекцию команд работы с матрицами, численного анализа и пр. (см. табл. 11.5). Команды пакета аналитических

преобразований Maple также доступны из среды MATLAB. Чтобы посмотреть их перечень, нужно выполнить команду

```
help toolbox\symbolic
```

Таблица 11.3. Разделы команд языка и среды MATLAB

Группа	Описание
ops	Операторы и специальные символы
lang	Языковые конструкции и команды отладки
datatypes	Типы данных и структуры
iofun	Низкоуровневые функции ввода-вывода
strfun	Команды обработки строк и символов
timefun	Команды работы со временем и датами

Таблица 11.4. Разделы графических команд

Группа	Описание
graphics	Графические команды
graph2d	Команды двумерной графики
graph3d	Команды трехмерной графики
specgraph	Специализированная графика
uitools	Команды пользовательского интерфейса

Таблица 11.5. Разделы математических команд

Группа	Описание
elfun	Элементарные математические функции
specfun	Специальные математические функции
elmat	Элементарные матрицы и преобразования матриц
funfun	Функции функций (минимизация, интегрирование и др.)
matfun	Матричные функции и линейная алгебра
polyfun	Команды интерполяции и работы с полиномами
datafun	Анализ данных и преобразование Фурье
sparfun	Команды работы с разреженными матрицами

Поскольку MATLAB имеет множество команд, то бывает трудно вспомнить точное написание команды. В таких случаях можно обратиться к справочной системе и организовать поиск команды с предполагаемым именем, скажем WORD, или воспользоваться командой

```
lookfor WORD
```

В результате поиска будет выведена информация о всех командах, где имеются включения данного слова.

MATLAB Help Desk содержит большой объем всевозможной информации. Многие из имеющихся в этой системе документов используют HTML-язык и доступны для просмотра через один из броузеров, например Netscape Communicator или Microsoft Explorer. Систему MATLAB Help Desk можно запустить из меню или набрать в строке ввода

```
helpdesk
```

В результате появится окно, представленное на рис. 11.5, воспользовавшись гиперссылками которого можно узнать новости данной реализации, организовать просмотр команд по темам, запустить поиск и т. д.

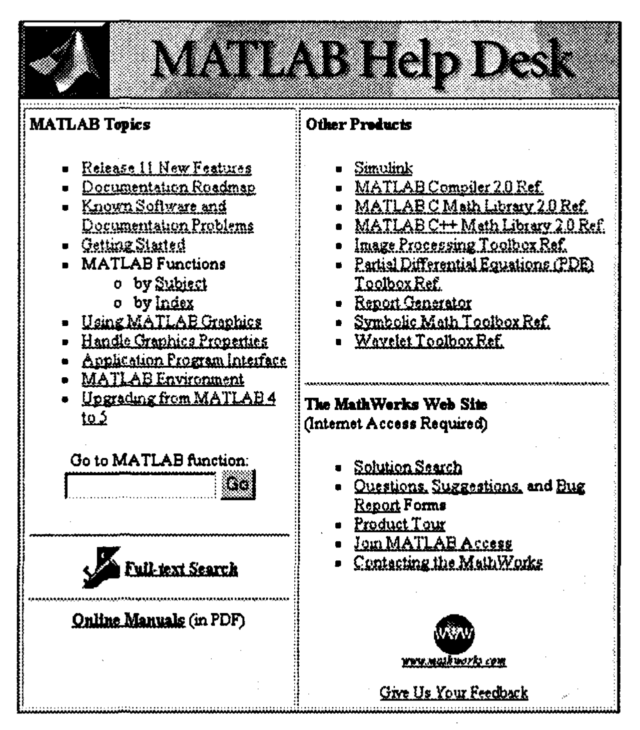


Рис. 11.5. Титульная страница Help Desk

В Help Desk описаны все команды и функции MATLAB, причем более детально, чем в справочной системе, а также даны примеры их использования. Имеются поисковая система и HTML-версии некоторых руководств из комплекта документации. Чтобы обратиться к документации для функции FUNCTION, достаточно в строке ввода основного окна набрать

```
doc FUNCTION
```

Комплект поставки MATLAB содержит справочную информацию, которую можно распечатать, и в виде PDF-файлов.

Кроме того, группа файлов-демонстраций MATLAB находится в подкаталоге \toolbox\matlab\demos. Для запуска примеров достаточно в строке ввода набрать

```
demo
```

Таблица 11.6. Список файлов-демонстраций

Имя файла	Назначение
demo.m	Меню демонстрации
bench.m	Оценка скорости используемого компьютера
fitdemo.m	Регрессионный анализ
fftdemo.m	Быстрое преобразование Фурье
membrane.m	Расчет прогиба упругой мембраны
meshdemo.m	Трехмерная графика на неравномерной сетке
matdemo.m	Основные операции с матрицами
nademo.m	Численные методы
odedemo.m	Решение обыкновенных дифференциальных уравнений
plotdemo.m	Графика

Интерфейс MATLAB 6.0

Новая версия MATLAB 6.0 сохраняет преемственность с предыдущими реализациями по языку, библиотекам команд, средствам отладки и пр. Изменения затронули интерфейс рабочего места (desktop), куда теперь включены средства работы с файлами, переменными и ассоциированными с MATLAB приложениями. При первом запуске рабочий стол выглядит так, как показано на рис. 11.6, хотя могут отличаться пункты в окне Launch Pad. Вид и расположение окон легко изменить по своему желанию.

Рассмотрим основные компоненты рабочего стола и укажем, что нового содержится в версии MATLAB 6.0.

- Командное окно, осуществляющее запуск команд и получающее результаты их выполнения. Здесь стало доступно контекстное меню для вывода выделенных переменных, открытия файлов-функций и получения справки по ним. Кроме того, конструкции языка MATLAB выделяются цветом.
- Список введенных в сеансе команд (History command) является новым инструментом, организующим просмотр, копирование и повторный запуск этих команд.
- Новым средством является Launch Pad, обеспечивающий быстрый доступ к инструментам (Workspace, Path, GUI Builder), справке и документации.
- Система справки обрела новый интерфейс, заменивший HelpDesk.
- Просмотр каталогов является новым средством, обеспечивающим работу с файлами, поиск и замену строк в файлах.
- Рабочая область (Workspace Browser) обеспечивает просмотр и внесение изменений в переменные, а также предоставляет доступ к графическому интерфейсу для считывания данных из бинарных и текстовых файлов Import Wizard.
- Редактор массивов (Array Editor) предназначен для просмотра и редактирования матриц, строк и массивов ячеек из строк, имеется возможность изменения формата представления данных.

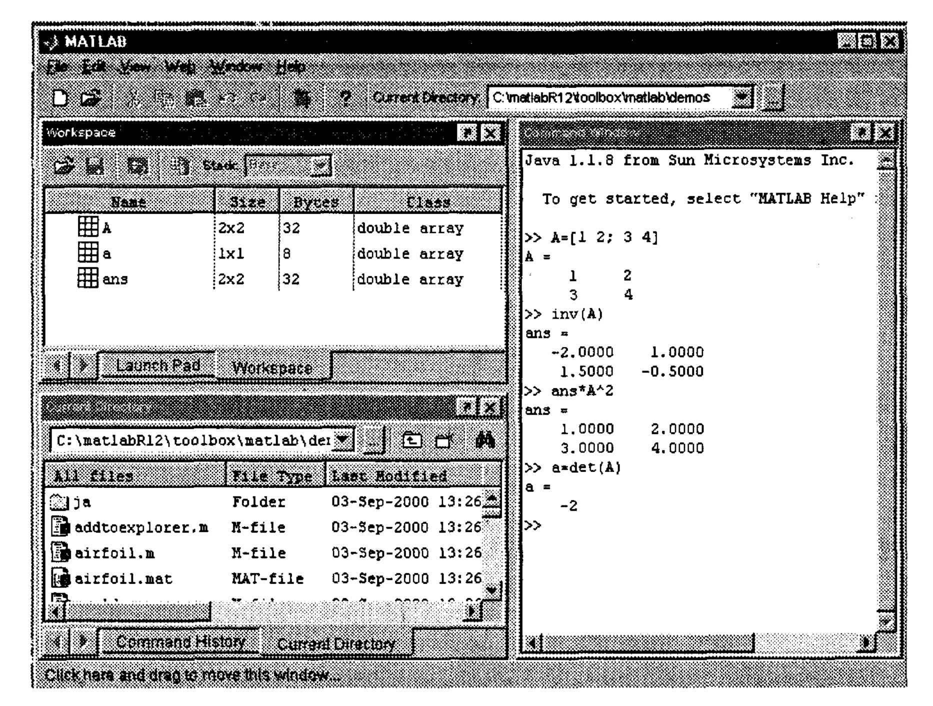


Рис. 11.6. Окно MATLAB 6.0

- Редактор-отладчик (Editor/Debugger) обеспечивает создание, редактирование и отладку m-файлов. Модернизация расширила возможности редактора (см. рис. 11.7), в частности добавлены следующие возможности: показ номеров строк, превращение массива строк в комментарии, изменение цветов для выделения синтаксических конструкций, поиск фразы в нескольких файлах, возможность восстановления файлов при неудачном завершении предыдущего сеанса, подсказки по данным, сохранение точек останова при записи файла.

Коротко о других нововведениях:

- окно Set Path дает доступ к каталогам и предоставляет новый интерфейс вместо Path Browser;
- профилер теперь поддерживает оценку времени выполнения файлов источников;
- добавилось новое средство Source Control Interface;
- включена поддержка Windows 2000 для документов Notebook.

Новым в системе справки является то, что почти вся документация подготовлена в виде HTML-файлов. Она лучше всего отражает текущее состояние системы MATLAB. Доступ к справочной информации реализуется при помощи просмотрщика (Browser, рис. 11.8), учитывающего специфику и организацию продуктов семейства MATLAB. Раскрываемые оглавления по системам, индексация по темам, расширенные поисковые возможности — все это позволяет легко найти нужную информацию. Новым является сохранение закладок и использование контекстного меню при нажатии правой кнопки мыши.

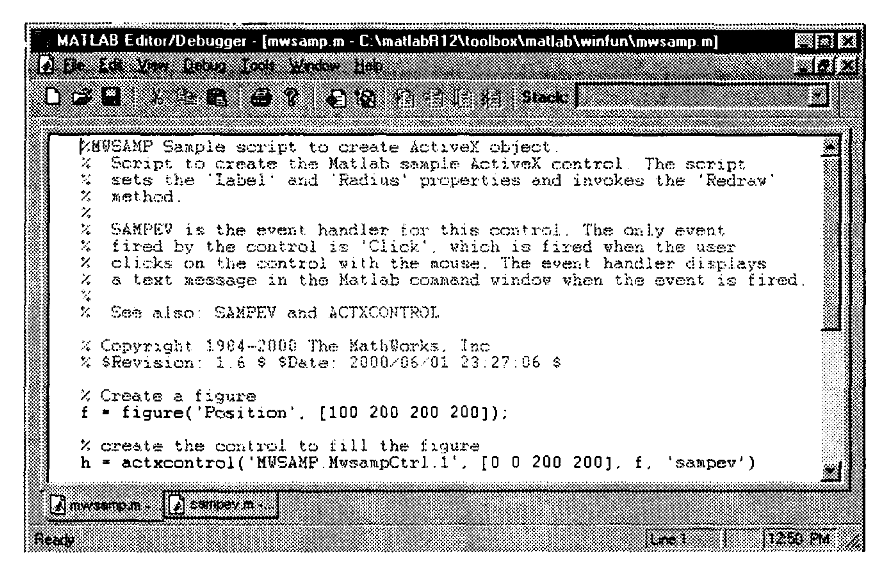


Рис. 11.7. Окно редактора

Рис. 11.8. Система справки

Для быстрого считывания в рабочую область двоичных или ASCII данных удобно пользоваться системой Import Wizard. Выберем в меню File пункт Import Data и в появившемся диалоговом окне откроем нужный файл. Система Import Wizard откроет файл и выведет обнаруженные переменные. Укажем нужные переменные и нажмем кнопку Finish для их считывания. Программа распознает данные различных

типов: изображения (.gif, .jpg, .jpeg, .pcx), таблицы (.csv, .xls, .wk1), звуковые (.wav, .au, .snd), аудио (.avi) и текстовые файлы (.txt, .dat, .dlm) с различными разделителями (табуляции, пробелы, запятые). Если указать разделитель, то будет прочитан файл с нетрадиционным разделителем.

Ряд команд поддерживает совместимость со старыми названиями. Например, по команде pathtool запускается диалоговое окно Set Path вместо системы Path Browser. Система Help browser загружается вместо Help Desk по командам helpdesk, helpwin и doc. В новых реализациях эти команды могут быть удалены.

Изменения коснулись в первую очередь интерфейса и ряда пакетов Toolboxes. В MATLAB 6 пакет Symbolic Math Toolbox основан на версии Maple V Release 5, в которой произведены усовершенствования ряда команд по сравнению с предыдущей реализацией.

Кроме того, расширились возможности компилятора MATLAB, для получения справки нужно набрать mcc -?. Добавились различные способы оптимизации кода, поддерживаются команды pause и continue. Библиотека MATLAB C/C++ Graphics Library позволяет использовать графические возможности MATLAB и интерфейс GUI для запускаемых независимо от MATLAB приложений, реализованных на C или C++. Значительно усовершенствована производительность всех функций математической библиотеки, в частности добавлены быстрые скалярные версии многих функций, графическая библиотека теперь поддерживает команды печати. В создаваемых программах можно использовать многие элементы MATLAB: текст, сетки, линии, многоугольники, компоненты графического интерфейса (меню, кнопки, диалоговые окна). Для создания таких приложений следует использовать компилятор MATLAB, вызов процедур MATLAB C/C++ Graphics Library из модулей C и C++ не поддерживается. Внесены изменения в библиотеку математических процедур MATLAB C/C++ Math Library (номер новой версии также 2.1).

При реализации новых возможностей среды MATLAB 6 использована технология Java. Однако поддержку Java можно отключить, для этого достаточно запустить программу matlab.exe с параметром -nojvm. В этом случае интерфейс аналогичен интерфейсу MATLAB 5.3. О других параметрах можно узнать из документации.

Элементы работы

В процессе сеанса пользователь вводит команды, которые исполняются, и результат выводится на дисплей (знаком (») помечены строки ввода команд MATLAB). Типичная команда присваивает переменной результат выполнения некоторого выражения, которое составляется из операторов, функций и имен переменных:

```
» VARIABLE = EXPRESSION
```

или в простейшем варианте

```
» EXPRESSION
```

Обработка выражения производится при нажатии клавиши ввода; на одной строке может быть несколько выражений, разделенных запятыми или точкой с запятой. Можно набирать длинные предложения, переходя на новые строчки, нажимая клавишу ввода (например, при вводе матрицы), или запустить m-файл на

выполнение при помощи пункта Run Script меню File. Если имя переменной и знак присваивания опущены, то результат выполнения выражения присваивается переменной ans.

Для редактирования вводимой команды могут быть использованы клавиши со стрелками и другие клавиши (Home, End, Delete). Исполненные команды заносятся в стек, и их можно вызвать при помощи клавиш со стрелками вверх и вниз, а затем отредактировать. При ошибке в задании команды раздается звуковой сигнал и выводится сообщение об ошибке.

Очистить переменную VAL можно командой

```
clear VAL
```

Команда clear без параметров очищает все переменные текущего сеанса. При работе может понадобиться очистить рабочую область и убрать «мусор» — для этого используется команда pack, которая сохраняет все переменные на диске, очищает память и снова загружает переменные.

Если потребуется остановить процесс вычислений, не покидая MATLAB, то здесь поможет обычная комбинация Ctrl+Break.

Для сохранения значений переменных текущего сеанса имеется команда save (пункт меню File/Save Workspace As...). Если не указано имя файла, то по умолчанию все значения будут записаны в файл matlab.mat. Восстановление всех переменных произойдет при следующем запуске пакета после исполнения команды load (пункт меню File/Load Workspace...). Для записи в файл команд и результатов имеется команда diary.

Каталоги MATLAB, в которых находятся функции пакета и данные (\work), перечислены в списке путей доступа Path Browser. Чтобы сделать доступной программу или функцию из какого-нибудь другого каталога, нужно добавить путь к этому каталогу.

Все расчеты в MATLAB выполняются с двойной точностью, а для представления чисел на экране имеются разные форматы. Нужный формат может быть определен в меню (File/Preferences) либо при помощи команды format. Существуют следующие способы представления чисел.

Таблица 11.7. Форматы вывода на экран

Формат	Представление
short	Число отображается с 4 цифрами после десятичной точки или в формате short e
short e	Число в экспоненциальной форме с мантиссой из 5 цифр и показателем из 3 цифр
bank	Число с любым количеством цифр до десятичной точки и двумя цифрами после
rat	Представление в виде рационального дробного числа
long	Число с 16 десятичными цифрами
long e	Число в экспоненциальной форме с мантиссой из 16 цифр и показателем из 3 цифр
hex	Число в шестнадцатеричной форме
+	Символическое отображение чисел (плюс — положительное число, минус — отрицательное и пробел для нуля)

Дадим примеры вывода числа a=12.3456789 в различных форматах:

```
» format short, a
a =
   12.3457
» format short e, a
a =
  1.2346e+001
» format bank, a
a =
         12.35
» format rat, a
a =
  1000/81
» format long, a
a =
  12.34567890000000
» format long e, a
a =
    1.234567890000000e+001
» format hex, a
a =
   4028b0fcd324d5a2
```

Команда format + дает компактный способ отображения структуры больших массивов. По умолчанию действуют формат short и стиль loose, который добавляет пустые строки при выводе результата. Стиль compact подавляет вывод пустых строк. Команда format без параметров восстанавливает значения по умолчанию.

12 ГЛАВА Элементы языка MATLAB

Изначально MATLAB был реализован на Фортране, и, хотя современные версии пишутся на C, язык MATLAB или М-язык конструкциями и отчасти синтаксисом напоминает Фортран. М-язык является языком высокого уровня и предоставляет достаточные возможности для реализации разнообразных вычислений, задач обработки данных и т. д. Этот язык сконструирован для решения математических задач и содержит специальные средства для эффективного выполнения математических операций.

В этом разделе даны общие сведения о работе с матрицами, арифметических и логических операциях, структурах и операторах MATLAB.

Синтаксис и данные

Переменные в MATLAB не нужно предварительно описывать, указывая их тип. Все данные хранятся в виде массивов: числовые переменные (внутренний тип `numeric`), текстовые строки (`char`), ячейки (`cell`) и структуры (`struct`), при помощи которых создаются пользовательские объекты (`user object`).

Числовые массивы состоят из комплексных чисел с двойной точностью (тип `double`) и могут храниться целиком или в упакованном виде в случае разреженной матрицы (тип `sparse`). Двумерный массив — это матрица, одномерный — вектор, а скаляр — матрица размера 1×1. Кроме того, существует несколько форматов записи данных в файлы (`int8`, `uint8`,...), отличающихся количеством используемых байтов. Схема типов MATLAB приведена на рис. 12.1.

Имя переменной должно начинаться с буквы, за ней могут идти буквы, цифры и символ подчеркивания. Допустимы имена любой длины, но MATLAB идентифицирует их по первым 31 символам и различает большие и малые буквы. В MATLAB имеется ряд констант, описанных в табл. 12.1. Отметим, что имя `NaN` (`Not-a-Number`) зарезервировано для результата операций `0/0`, `0*inf`, `inf-inf` и т. п. Информацию об имеющихся константах можно получить, вызвав справку о разделе команд `elmat`.

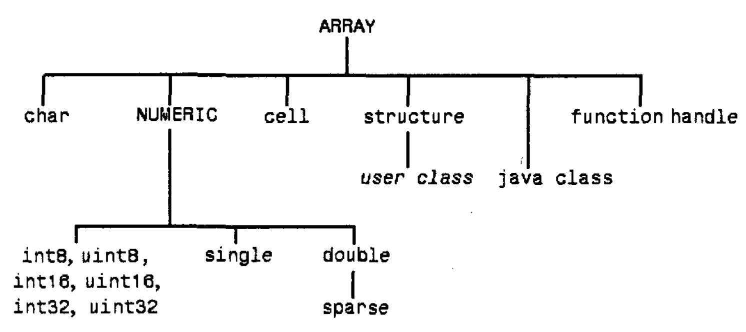


Рис. 12.1. Типы MATLAB

Буквы i и j можно использовать как обычные переменные, например в качестве счетчика цикла. В этом случае мнимую единицу можно задать заново:

```
» jj=sqrt(-1)
jj =
       0 + 1.0000i
```

Заметим, что для обозначения мнимой единицы применяется символ i. Выражения и команды в MATLAB записываются с помощью различных символов, краткое описание которых дано в табл. 12.2.

Таблица 12.1. Зарезервированные имена констант

Имя	Описание
ans	Результат последней операции
i, j	Мнимая единица
pi	Число π
eps	Машинная точность
realmax	Максимальное вещественное число
realmin	Минимальное вещественное число
inf	Бесконечность
NaN	Нечисловая переменная
end	Наибольшее значение индекса размерности массива

Приведем пример использования ряда зарезервированных слов и специальных символов для формирования матрицы размера 2×3:

```
» A=[i*realmax, pi:5; -realmin+j*eps -inf inf/Inf] % пример
A =
              0 +1.7977e+308i      3.1416e+000     4.1416e+000
  -2.2251e-308 +2.2204e-016i      -Inf                    NaN
```

В этом примере точка с запятой отделяет строки матрицы, в качестве мнимой единицы фигурируют символы i и j, а в правой части строки знаком % отмечено начало комментария. Для вывода матрицы на экран использован формат short e, а если задать формат short, то матрица A будет выглядеть следующим образом:

```
» format short, A
A =
1.0e+308 *
        0 + 1.7977i   0.0000            0.0000
   0.0000 +               -Inf             NaN
```

Обратим внимание, что второй и третий элементы первой строки и первый элемент второй строки не равны нулю, а просто произошло округление при выводе матрицы на экран.

Таблица 12.2. Специальные символы

Символ	Назначение
[]	Квадратные скобки используются при задании матриц и векторов
	Пробел служит для разделения элементов матриц
,	Запятая применяется для разделения элементов матриц и операторов в строке ввода
;	Точка с запятой отделяет строки матриц, а точка с запятой в конце оператора (команды) отменяет вывод результата на экран
:	Двоеточие используется для указания диапазона (интервала изменения величины) и в качестве знака групповой операции над элементами матриц
( )	Круглые скобки применяются для задания порядка выполнения математических операций, а также для указания аргументов функций и индексов матриц
.	Точка отделяет дробную часть числа от целой его части, а также применяется в составе комбинированных знаков (.*, .^, ./, .\)
...	Три точки и более в конце строки отмечают продолжение выражения на следующей строке
%	Знак процента обозначает начало комментария
!	Восклицательный знак отмечает начало команды MS DOS, например команда !dir выводит оглавление текущего каталога (директории)
'	Апостроф указывает на символьные строки, а для включения самого апострофа в символьную строку нужно поставить два апострофа подряд

Задание матриц

По умолчанию все числовые переменные в MATLAB считаются матрицами с комплексными числами, так что скалярная величина есть матрица первого порядка, а векторы являются матрицами, состоящими из одного столбца или одной строки. Матрицу можно ввести, задав ее элементы или считав данные из файла, а также в результате обращения к стандартной или написанной пользователем функции. Элементы матрицы в пределах строки отделяются пробелами или запятыми, поэтому при задании числа в экспоненциальной форме (мантисса и порядок степени) никакие пробелы не допускаются. Матричные данные размещаются в памяти последовательно по столбцам.

Непосредственное задание матрицы можно осуществить несколькими способами. Например, вектор-столбец, то есть матрица, вторая размерность которой равна единице, может быть присвоена переменной a вводом одной строки:

```
» a=[7e-6+5i; 4; 3.2e1]          % Ввод вектора-столбца
a =
   0.0000 + 5.0000i
   4.0000
  32.0000
```

или вводом нескольких строк:

```
» a=[                            % Ввод вектора по строкам
7e-6+5i
4
3.2e1];
```

В качестве примера задания вектора при помощи функций MATLAB приведем команды `linspace` и `logspace`. Они позволяют создавать векторы со значениями, меняющимися соответственно в арифметической и геометрической прогрессии. Три параметра команды `linspace` задают соответственно первый и последний члены арифметической последовательности, а также число членов. Например:

```
» linspace(1,-2,4)
ans =
     1     0    -1    -2
```

Параметры команды `logspace` задают соответственно десятичные порядки первого и последнего членов геометрической прогрессии, а также число членов. Например:

```
» logspace(1,-1,5)
ans =
   10.0000    3.1623    1.0000    0.3162    0.1000
```

Векторы могут быть сформированы как диапазоны — при помощи двоеточий, разделяющих стартовое значение, шаг и предельное значение. Если величина шага отсутствует, то по умолчанию его значение равно единице:

```
n:m:k
```

В результате будет сформирован вектор, последний элемент которого не больше `k` для положительного шага `m`, и не меньше — для отрицательного:

```
[n,n+m,n+m+m,...]
```

Например:

```
» a=1:-2:-4
a =
     1    -1    -3
```

Задание диапазона используется также при организации цикла, смотрите раздел «Элементы программирования» этой главы.

Большую матрицу можно определить поэлементно при выполнении оператора цикла или набрать ее в обычном редакторе в виде ASCII-файла (например, **data.ext**), а затем считать при помощи команды

```
» load data.ext
```

В результате матрица будет присвоена переменной `data`.

Для задания ряда матриц специального вида имеются команды MATLAB, аргументами которых являются размерности создаваемых матриц. Если указано одно число N, то формируется квадратная матрица N×N. Например, команда rand(n,m) создаст случайную матрицу размерности n*m, а команда hilb(n) определит гильбертову матрицу n-го порядка. Кроме того, для задания матрицы можно написать свою команду — функцию, сохранив ее в виде m-файла (см. далее).

Таблица 12.3. Функции описания матриц

Имя	Назначение
eye	Единичная матрица
zeros	Нулевая матрица
ones	Матрица из единиц
rand	Случайная матрица со значениями из интервала [0,1]
hilb	Гильбертова матрица
magic	Матрица магического квадрата
diag	Создание диагональной матрицы или выделение диагонали
triu	Выделение верхней треугольной части матрицы
tril	Выделение нижней треугольной части матрицы

Определение теплицевой матрицы имеет особый формат:

```
» T = toeplitz([1 2 3], [1 4 6])
T =
     1     4     6
     2     1     4
     3     2     1
```

В MATLAB имеется возможность, обратившись к команде gallery, получить матрицу из подготовленного разработчиками набора стандартных матриц. Чтобы посмотреть список матриц, достаточно набрать

```
help gallery
```

Чтобы вызвать нужную матрицу, нужно обратиться к функции gallery

```
[out1,out2,...]=gallery(NAME,PAR1,PAR2,...)
```

Здесь NAME имя матрицы из списка gallery, а PAR1, PAR2 — дополнительные параметры. Например, разреженная матрица для задачи Пуассона при сетке 6×6 будет получена по команде:

```
» Poi=gallery("poisson",6); size(Poi)
ans =
    36    36
```

Из стандартных блоков можно определить новую матрицу:

```
» B=[triu(ones(2)),zeros(2,1);ones(1,2),eye(1)]
B =
     1     1     0
     0     1     0
     1     1     1
```

В результате получается матрица размерности 3×3. Напомним, что при задании матрицы запятая отделяет элементы строки, а точка с запятой разделяет строки.

Обращение к элементам матрицы

Обращение к элементу матрицы производится по естественному правилу, — в круглых скобках после имени матрицы даются индексы, которые должны быть положительными целыми числами. Например, A(2,1) означает элемент из второй строки первого столбца матрицы A.

Для дальнейших примеров введем матрицу 2×2:

```
» A=[1 2+5*i; 4.6e-7 3];
```

Если в качестве индекса задать комплексное число с дробной вещественной частью, то MATLAB выведет предупреждение, отбросит мнимую составляющую, произведет округление дроби и попытается выполнить операцию:

```
» A(3/2+4*i)
Warning: Complex part of array subscript is ignored.
Warning: Subscript indices must be integer values.
ans =
  4.6000e-007
```

Данный пример показывает, что числа хранятся по столбцам и при обращении к данному двумерному массиву элемент A(2) есть то же самое, что и A(2,1). Чтобы изменить элемент матрицы, ему нужно присвоить новое значение:

```
» A(2,3)=sin(1)                       % Третий элемент второй строки
A =
   1.0000            2.0000 + 5.0000i        0
   0.0000            3.0000              0.8415
```

Заметим, что изначально матрица A состояла из двух строк и столбцов. Расширение матрицы (добавление третьего элемента во вторую строку) не потребовало никаких дополнительных действий, при этом третий элемент в первой строке был обнулен автоматически.

Если обратиться к отсутствующему элементу матрицы, то будет выведено сообщение об ошибке:

```
» A(3,1)^2                            % У матрицы только две строки!
Index exceeds matrix dimensions.
```

Размер матрицы можно уточнить по команде size, а результат команды size можно использовать для организации новой матрицы. Например, нулевая матрица того же порядка, что и матрица A, будет сформирована по команде

```
» A2=zeros(size(A))
A2 =
     0     0     0
     0     0     0
```

Для преобразования матрицы в матрицу с другим числом строк и столбцов служит команда reshape

```
» A3=reshape(A2,3,2)
A3 =
     0     0
     0     0
     0     0
```

С помощью двоеточия легко выделить часть матрицы. Например, вектор из первых двух элементов третьего столбца матрицы A задается выражением

```
» A(1:2,3)
ans =
         0
    0.8415
```

Двоеточие само по себе означает строку или столбец целиком. Работа с индексами в MATLAB очень удобна. Например, чтобы выделить несколько столбцов массива, достаточно организовать вектор из номеров столбцов. Это можно сделать явным перечислением, а можно указать нужный диапазон. Для выделения матрицы, составленной из нечетных столбцов матрицы A, используем команду

```
» A(:,1:2:3)
ans =
    1.0000         0
    0.0000    0.8415
```

Здесь конструкция 1:2:3 означает изменение второго индекса от единицы до трех с шагом два. Двоеточие применяется также для замещения элементов матрицы. Следующая команда позволяет переставить первую и вторую строки матрицы A:

```
» A([1,2],:)=A([2,1],:)
A =
   0.0000              3.0000              0.8415
   1.0000              2.0000 + 5.0000i         0
```

Здесь в качестве индекса выступают векторы [1,2] и [2,1], а для их оформления использованы равносильные разделители: запятая и пробел. Для удаления элемента вектора достаточно присвоить ему пустой массив — пару квадратных скобок []. Чтобы вычеркнуть одну или несколько строк (столбцов) матрицы нужно указать диапазон удаляемых строк (столбцов) для одной размерности и поставить двоеточие для другой размерности. Например, для удаления двух последних столбцов матрицы A достаточно ввести команду

```
» A(:,2:end)=[]                          % вырезание строк
A =
    0.0000
    1.0000
```

Обратим внимание, что вместо числового значения индекса указано зарезервированное имя end — максимальное значение индекса.

В списке аргументов size второй параметр позволяет определить соответствующую размерность матрицы, например найти число столбцов матрицы. Для нахождения длины вектора можно воспользоваться также командой length. Число столбцов матрицы A1 равно 3, не зависимо от того, каким способом пользоваться:

```
» [size(A1,2), length(A1(1,:))]
```

```
ans =
     3     3
```

Вместо двоеточия можно также использовать функцию-синоним colon.

Арифметические операции

Набор арифметических операций в MATLAB состоит из стандартных операций сложения-вычитания, умножения-деления, операции возведения в степень и дополнен специальными матричными операциями. Запись операций сложения (вычитания) и умножения матриц естественна. Если операция применяется к матрицам, размеры которых не согласованы, то будет выведено сообщение об ошибке. Для поэлементного выполнения операций умножения, деления и возведения в степень применяются комбинированные знаки (точка и знак операции). Например, если за матрицей стоит знак (^), то она возводится в степень, а комбинация (.^) означает возведение в степень каждого элемента матрицы. При умножении (сложении, вычитании, делении) матрицы на число соответствующая операция всегда производится поэлементно.

Таблица 12.4. Знаки операций

Символ	Назначение
+, -	Символы плюс и минус обозначают знак числа или операцию сложения и вычитания матриц, причем матрицы должны быть одной размерности
*	Знак умножения обозначает матричное умножение; для поэлементного умножения матрицы применяется комбинированный знак (.*)
'	Апостроф обозначает операцию транспонирования (вместе с комплексным сопряжением); транспонирование без вычисления сопряжения обозначается при помощи комбинированного знака (.')
/	Левое деление
\	Правое деление
^	Оператор возведения в степень; для поэлементного возведения в степень применяется комбинированный знак (.^)

Проиллюстрируем различие обычного и поэлементного умножений при помощи следующего примера. Введем матрицу H размера 2×2 и матрицу O из единиц той же размерности:

```
» H=[0 1; 2 3], O=ones(size(H))
H =
     0     1
     2     3
O =
     1     1
     1     1
```

Перемножим матрицы, используя обычное умножение:

```
» H*O
ans =
```

```
    1    1
    5    5
```

Теперь применим поэлементную операцию:

```
» H.*O
ans =
    0    1
    2    3
```

Понятно, что при поэлементном умножении вектора b=[1,2,3] на себя (b.*b) или при поэлементном возведении в квадрат (b.^2) результат получается одинаковым:

```
[1 4 9]
```

Образуем матрицу умножением вектора-столбца, полученного транспонированием из строки, на исходный вектор-строку:

```
» C=b'*b
C =
    1    2    3
    2    4    6
    3    6    9
```

Прибавим к матрице C единичную матрицу той же размерности, умноженную на комплексное число π+i, и вычтем из полученного результата число 2:

```
» D=C+(pi+i)*eye(size(C))-2
D =
  2.1416 + 1.0000i        0               1.0000
       0             5.1416 + 1.0000i    4.0000
  1.0000             4.0000              10.1416 + 1.0000i
```

Для вычитания числа из матрицы оно заменяется матрицей нужного размера, все элементы которой равны данному числу, а уже затем вычисляется разность матриц.

Операция транспонирования для матрицы D дает следующий результат:

```
» D'
ans =
  2.1416 - 1.0000i        0               1.0000
       0             5.1416 - 1.0000i    4.0000
  1.0000             4.0000              10.1416 - 1.0000i
```

Если при сложении и умножении размеры матриц не соответствуют, то будет выведено сообщение об ошибке. Например:

```
» ones(2)*eye(3)
??? Error using ==> *
Inner matrix dimensions must agree.
```

Матричное левое деление может быть пояснено на примере системы линейных алгебраических уравнений Ax=b. Для ее решения достаточно набрать

```
x=A\b
```

Соответственно для системы xA=b будет получено решение (если оно есть) при помощи операции правого деления

```
x=b/A
```

При левом делении в случае квадратной матрицы используется метод Гаусса. Если матрица A не квадратная, то применяется факторизация Хаусхолдера с обнулением столбцов, а решение для недоопределенных или переопределенных систем находится методом наименьших квадратов. Правое деление может быть оформлено в терминах левого деления:

```
b/A=(A'\b')'
```

Представим пример для введенных ранее матрицы C и вектора b:

```
» b\C
ans =
    1.0000
    2.0000
    3.0000
```

Логические операции

Операторы отношения и логические операторы, а также соответствующие им команды позволяют проводить сравнения массивов одинакового размера. Результатом таких операций являются матрицы из нулей и единиц, причем единица означает истинность, а нуль — ложь.

Таблица 12.5. Операции отношения

Символ	Назначение	Имя функции
<	Меньше	lt
>=	Больше или равно	ge
>	Больше	gt
<=	Меньше или равно	le
==	Равно	eq
~=	Не равно	ne

При попытке сравнения векторов или матриц различной размерности будет выведено сообщение об ошибке. При сравнении скаляра с матрицей сначала из скалярной переменной создается матрица нужного размера, и уже затем происходит сравнение.

Операции (==, ~=) проводят сравнение вещественных и мнимых частей комплексных чисел, а операции (>, <, >=, <=) — только вещественных частей.

Таблица 12.6. Логические операции

Символ	Назначение	Имя команды
&	Логическое «и»	and
|	Логическое «или»	or
~	Отрицание	not
	Исключительное «или»	xor

Для логических операций ненулевое число отождествляется с единицей.

Приведем примеры:

```
» a=[1 2; 3 4]                         % ввод матрицы
a =
     1     2
     3     4
» b=2;                                 % ввод скаляра
» c=a>b                                % результат сравнения - матрица
c =
     0     0
     1     1
» a~=c                                 % сравнение матриц - матрица
ans =
     1     1
     1     1
```

Логические операции можно записывать в виде функций. Так, последнее сравнение представимо в виде:

```
» ne(a,c)                              % функция сравнения матриц
```

Необходимость логических операций возникает, например, в условных операторах языка MATLAB. При этом арифметические операции всегда имеют приоритет по отношению к логическим.

Элементарные логические операции дополнены набором функций MATLAB, позволяющих проверить для матриц некоторые условия, см. табл. 12.7. Проверка для матриц проводится по столбцам. Результатом проверки для вектора является число.

Таблица 12.7. Команды проверки и сравнения

Имя	Назначение
find	Поиск значений согласно заданному условию; определение индексов
all	Проверка того, что все элементы не равны нулю
any	Проверка того, что хотя бы один элемент не равен нулю
isempty	Выявление пустого массива
isequal	Проверка равенства матриц
issparse	Проверка матрицы на разреженность
nonzeros	Вывод ненулевых элементов массива
finite	Выявление ограниченных элементов массива
isnumeric	Проверка, является ли массив числовым
isinf	Выявление бесконечных элементов массива
isnan	Выявление элементов нечислового типа
isletter	Проверка на символ
isstr	Проверка на строковую переменную
isglobal	Проверка на глобальную переменную
strcmp	Сравнение двух строк

Проиллюстрируем некоторые из этих команд на следующих примерах.

Проверим массив c на наличие ненулевых элементов:

```
» any(c')
ans =
     0     1
```

Ряд функций дает возможность выявить пустую переменную (isempty), идентичность матриц (isequal), ответить, числовой ли это массив (isnumeric):

```
» [isnumeric(c) isempty([]) isempty(c) isequal(a,c)]
ans =
     1     1     0     0
```

Наконец, есть команда find, которая выводит индексы и значения согласно заданному условию. Например, для массива c, полученного ранее, команда поиска элементов, тождественно равных единице, выведет:

```
» d=find(c==1)
d =
     2
     4
```

Заметим, что результат содержит ссылки на два единичных элемента массива c, при этом используется одномерная индексация. Напомним, что матричные данные размещаются в памяти последовательно по столбцам. Действительно,

```
» c(d(2))-c(4)
ans =
     0
```

Приведем еще пример выделения нужных данных из массива. Рассмотрим задачу нахождения простых чисел на интервале [100,130]. В MATLAB имеется команда isprime, проверяющая число на простоту:

```
» A=100:130; P=A(find(isprime(A)))
P =
   101   103   107   109   113   127
```

Отметим, что команда find возвращает индексы элементов массива, удовлетворяющих некоторому условию, поэтому для вывода простых чисел из заданного диапазона нужно результат поиска подставить в качестве индекса.

Текстовые строки

Символы и текстовые строки в MATLAB вводятся при помощи простых кавычек. Во внутреннем представлении символы даны целыми числами. Конвертировать массив символов в числовую матрицу позволяет команда double. Обратная операция совершается по команде char. Печатаемые символы из стандартного набора ASCII представлены числами от 32 до 255. Объединить текстовые строки можно, просто заключив их в квадратные скобки или при помощи команды strcat. Приведем примеры для данных команд. Вначале введем строку:

```
» s='Привет'
s =
Привет
```

Отметим, что для ввода русских букв следует выбрать в меню File/ Preferences/ Command Windows Font шрифт с русской кодировкой. Теперь найдем коды введенных символов:

```
» v=double(s)
v =
   207   240   232   226   229   242
```

Объединим две строки:

```
» h=[v " от MATLAB"]
h =
Привет от MATLAB
```

Тот же результат получится, если вместо переменной v использовать строковую переменную s. В завершение попробуем применить команду strcat:

```
» strcat(s,v)
Warning: Input must be a string.
>In C:\MATLABR11\toolbox\matlab\strfun\deblank.m at line 15
 In C:\MATLABR11\toolbox\matlab\strfun\strcat.m at line 57
ans =
ПриветПривет
```

Действительно, переменная v не является строкой, однако, выведя предупреждение, MATLAB попытался выполнить запрашиваемое соединение переменных. В результате получилась текстовая строка.

Для перевода численных данных в строковые переменные имеется ряд команд преобразования. В табл. 12.8 приведены некоторые команды для этих и обратных операций, а полный список можно получить по команде

```
help strfun
```

Таблица 12.8. Команды преобразования строка/число

Имя команды	Действие
num2str	Перевод числа в строку
int2str	Перевод целого числа в строку
mat2str	Преобразование матрицы в строку
str2mat	Объединение строк в матрицу
str2num	Преобразование строки в число
strcat	Объединение строк

Приведем простой пример:

```
» z=1.2e3; strcat("z=",num2str(z))
ans =
z=1200
```

Многомерные массивы

Массивы с числом размерностей более двух считаются многомерными. Такие массивы могут быть считаны из файла или созданы при помощи команд, таких как

zeros, ones, rand. Число параметров при обращении к этим командам должно соответствовать размерности вводимого массива. Например, трехмерный массив 2×4×2 из нулей будет организован по команде

```
» S=zeros(2,4,2);
```

Обращение к элементам многомерного массива производится по обычным правилам работы с массивами, так что действуют двоеточия для указания диапазона, a end означает максимальное значение данной размерности. Чтобы изменить какой-нибудь элемент, достаточно присвоить ему значение точно так же, как для обычных массивов, а если элемента не было, то произойдет увеличение размерности массива:

```
» S(3,1,2)=13; size(S(:,:,1))
ans =
     3     4
```

Заметим, что пополнение массива означает дополнительные затраты времени на переписывание данных. Поэтому для повышения скорости расчета рекомендуется описывать максимальную размерность массива сразу (резервировать память), если это возможно. Когда многомерный массив организуется для хранения нескольких матриц одинакового размера, то первые два индекса удобнее отвести под строки и столбцы матриц, а последний индекс — для номера матрицы. Тогда в результате получим:

```
» S2=S(:,:,end)
S2 =
     0     0     0     0
     0     0     0     0
    13     0     0     0
```

Напомним, что резервирование памяти ускоряет работу в среде MATLAB, поскольку не расходуется время на пополнение массивов.

Массивы ячеек

Для хранения разнородных объектов (массивов разных размерностей, разнотипных данных) удобно пользоваться массивами ячеек, которые создаются двумя способами: по команде cell или заключением объектов в фигурные скобки. Например:

```
» C = {sum(S) min(max(S)) sum(sum(sum(S)))}
C =
    [1x4x2 double]    [1x1x2 double]    [13]
```

Для указания элементов массива ячеек используются фигурные скобки, так что в результате обращения к третьему элементу массива C получим число 13:

```
» C{3}
ans =
    13
```

а содержимое первого элемента массива ячеек C есть:

```
» C1=C{1}
C1(:,:,1) =
     0     0     0     0
```

```
C1(:,:,2) =
   13     0     0     0
```

Для превращения структуры C1 в обычный массив можно воспользоваться командами squeeze или shiftdim, которые удаляют равные единице размерности (матрица в один столбец или одну строку превращается в вектор):

```
» C2=shiftdim(C1)'
C2 =
    0     0     0     0
   13     0     0     0
```

Нужно помнить, что составные части массива ячеек представлены копиями, так что при изменении исходного массива S в объекте C никаких изменений не последует. Иными словами, это не указатель на массив, а сам массив.

Для преобразования массива символов в массив ячеек применяется команда cellstr, а обратная процедура реализуется командой char.

Если потребуется создать текст из нескольких строк, то обычный прием отделения строк точкой с запятой может не сработать, поскольку все строки должны быть одной длины.

В таких случаях можно подготовить массив ячеек:

```
» C={"One";"Three";"Seven"}
C =
    "One"
    "Three"
    "Seven"
```

и затем преобразовать его в массив символов при помощи команды char:

```
» S=char(C)
S =
One
Three
Seven
```

Для получения строк одной длины MATLAB добавляет нужное число пробелов, что легко увидеть, затребовав данные о размерностях массива S:

```
» size(S)
ans =
    3     5
```

Структуры

Структурами MATLAB (тип struct) являются многомерные массивы. Доступ к ним осуществляется путем указания индексов-имен. Например, можно создать скалярную структуру из двух полей:

```
» S.name='трап'; S.order=2
S =
     name: "трап"
    order: 2
```

Расширение структуры производится по тому же правилу, что и добавление строк или столбцов в массив:

```
» S(2).name='Симп'; S(2).order=4;
```

Для пополнения структуры можно также использовать специальную команду struct. Организуем третий элемент структуры S, присвоив полю name значение прям, а полю order — значение 2.

```
» S(3)=struct("name".'прям'.'order'.2)
S =
1x3 struct array with fields:
    name
    order
```

Чтобы вывести содержимое отдельных полей структуры, нужно использовать фигурные либо квадратные скобки в зависимости от того, символьные или числовые данные связаны с данным полем:

```
» {S.name}
ans =
    "трап"    "Симп"    "прям"
» [S.order]
ans =
     2     4     2
```

Преобразование данных в массив символов осуществляется по команде char:

```
» char(S.name)
ans =
трап
Симп
Прям
```

Элементы программирования

Система MATLAB включает стандартный набор конструкций языка программирования высокого уровня. В этом разделе описана работы с условными операторами, оператором выбора, циклами, а также программирование собственных функций.

Условные операторы и циклы

Простейший условный оператор имеет вид:

```
if BOOL. EXPR. end
```

Команды EXPR будут выполнены, если истинно условие BOOL.

Оператор «или» дает альтернативу:

```
if BOOL. EXPR. else EXPR_2. end
```

Напомним, что запятая (или точка с запятой) после условия BOOL необязательна, если команды EXPR расположить на следующей строке. При добавлении еще одного служебного слова — elseif — условный оператор становится мощнее:

```
if BOOL_1, EXPR_1
elseif BOOL_2, EXPR_2
elseif BOOL_3, EXPR_3
. . .
else EXPR_0
end
```

Здесь проверки следуют одна за другой. В том случае, если условие BOOL_N истинно, будет выполнен оператор EXPR_N. Например, при x=4 в результате выполнения условного оператора:

```
» if x<0, 0, elseif x<3, 3, elseif x<5, 5, else 13, end
```

получим:

```
ans =
    5
```

Оператор-переключатель switch предоставляет возможность разветвления и выполнения операторов в зависимости от значения переменной VAR (скаляра или строки):

```
switch VAR
case VAR1, EXPR_1
case {VAR2, VAR3,...}, EXPR_2,
otherwise, EXPR_0
end
```

Здесь VAR1, VAR2, VAR3, ... — различные значения, которые может принимать переменная VAR, а EXPR_1, EXPR_2 и т. д. — группы операторов. Переключение возможно по единственному значению (выбор VAR1) и по группе значений (выбор из VAR2, VAR3, ...). Кроме того, если осознанного выбора не сделано, то выполняются операторы группы EXPR_2 (ключевое слово otherwise).

Приведем пример со строковой переменной, здесь команда disp используется для вывода текстового сообщения:

```
» METHOD='Симп';
switch METHOD
   case {"трап",'прям'}, disp("Метод 2 порядка")
   case "Симп", disp("Метод 4 порядка")
   otherwise, disp("Метод неизвестен")
end
```

В результате получим:

```
Метод 4 порядка
```

Перечислительный цикл имеет вид:

```
for N=N0:DN:N1, EXPR end
```

Здесь N — счетчик, пробегающий с шагом DN значения от N0 до N1 (учитываются вещественные части чисел); а EXPR обозначает выполняемые в теле цикла команды.

Цикл с условием:

```
while BOOL, EXPR end
```

выполняется до тех пор, пока не будет нарушено условие BOOL. Для выхода из тела цикла имеется оператор прерывания break. В MATLAB 6 появился оператор continue, означающий пропуск операторов и переход к следующему значению переменной цикла.

Приведем примеры. В результате выполнения команд

```
» x=[]; for k=1:5; if k==5, break, end, x=[x,k^3]; end;
```

будет сформирован вектор:

```
[1,8,27,64]
```

Здесь вначале организована пустая переменная x, к которой в процессе выполнения цикла добавляются новые элементы. К тому же результату приведет выполнение цикла с условием:

```
» x=1; k=1; while k<4; k=k+1; x=[x,k^3]; end;
```

Подчеркнем, что матричные операции MATLAB выполняет быстрее, чем действия с циклами. Формирование массива x потребует меньше времени, если не пользоваться оператором цикла:

```
» k=1:4; x=k.^3;
```

Хотя для задания гильбертовой матрицы имеется функция hilb, приведем пример ее определения с помощью следующих двух циклов:

```
for k=1:3
        for m=1:4
            A(k,m)=1/(k+m-1);
        end
end
```

Если эти циклы вводить в командном окне, то переход на следующую строку происходит при нажатии клавиши **ВВОД**. При этом MATLAB не приступит к выполнению цикла, пока не оформится конструкция цикла.

Для обработки ошибок вычислений в MATLAB предусмотрена конструкция

```
try OPER_1 catch OPER_2 end
```

Здесь обозначения OPER_1 и OPER_2 представляют группы операторов. Если фрагмент программы содержит данную конструкцию и при выполнении операторов OPER_1 не произошло ошибки, то управление передается операторам, стоящим за словом end. Если же возникает ошибка, то служебной переменной lasterr присваивается сообщение о характере ошибки и выполняется группа операторов OPER_2.

Следующий пример показывает, что MATLAB считает ошибкой:

```
» try, for k=0:1,k/0,end, fi, catch, ERR=lasterr, end,'Fi'
Warning: Divide by zero.
ans =
   NaN
Warning: Divide by zero.
ans =
   Inf
ERR =
Undefined function or variable "fi".
```

```
ans =
Fi
```

В заключение отметим, что обработка циклов интерпретатором MATLAB замедляет скорость расчетов, поэтому по возможности следует применять векторизацию — проведение операций по строкам или по колонкам с использованием двоеточий для указания обрабатываемых диапазонов. Сравним затраты времени для вычисления синуса от последовательности чисел, используя оператор цикла:

```
» tic, for k=1:23456, y=sin(k);end, toc
elapsed_time =
   1.0400
```

и расчета с применением векторизации:

```
» tic, k=1:23456; y=sin(k); toc
elapsed_time =
   0.0600
```

Второй способ потребовал значительно меньше времени. Полезно также предварительно выделять память, чтобы исключить переписывание наращиваемых массивов. Приведем пример, в котором массив из тысячи элементов в цикле заполняется значениями счетчика цикла. Если не описать переменную B как массив, то получается следующий результат:

```
» n=1000;tic,for k=1:n, B(k)=k; end; toc
elapsed_time =
   0.4400
```

Если же ввести предварительно массив из тысячи элементов, то экономия очевидна:

```
» n=1000;A=zeros(1,n);tic,for k=1:n, A(k)=k; end; toc
elapsed_time =
   0.0600
```

Вообще, при разработке программ полезно использовать профилер MATLAB, чтобы выяснить, какие части программы вызывают особые затраты времени, а затем оптимизировать их. В частности, для этого можно использовать mex-файлы — написанные на C или Фортране модули и откомпилированные так, чтобы их можно было вызывать из MATLAB. О профилере и создании mex-файлов речь пойдет в главе 16 «Программирование в MATLAB».

Функции и файлы-источники (m-файлы)

Для простых операций удобен интерактивный режим, но если вычисления нужно многократно повторять или необходимо реализовать сложные алгоритмы, то следует использовать m-файлы MATLAB. Существуют два вида m-файлов — файлы внешних функций и script-файлы (последовательности команд или программы). Записанные в script-файл команды будут исполнены, если в командной строке ввести имя script-файла. Переменные script-файла являются глобальными. Их значения заместят значения таких же переменных, которые были использованы до вызова данного script-файла.

Примером script-файла является файл first.m, приведенный в главе 11 «Работа в MATLAB»:

```
A=[1 2; 3 4]
inv(A)
ans*A^2
a=det(A)
```

В отличие от script-файлов обращение к внешней функции обычно сопровождается передачей аргументов. Внутренние переменные по умолчанию являются локальными, и для того, чтобы использовать их как глобальные, требуется соответствующее описание. По завершении работы функции выводится результат — некоторое число матриц, строк и т. д. Схематично структура функции имеет вид:

```
function [OU1,OU2,...]=FUN(IN1,IN2,...)
%COMMENT
global G1 G2 ...
OPERATORS
```

Здесь OU1, OU2, ... — выходные параметры (результаты), IN1, IN2, ... — входные параметры, %COMMENT — комментарий, G1, G2, ... — глобальные переменные, OPERATORS — операторы, составляющие тело функции. Обычно выход из функции происходит после выполнения последнего оператора, если же нужно завершить работу раньше, то используется команда return. Приведем пример. В текстовом редакторе подготовим файл **func.m** из двух строк:

```
function y=func(x);
                                          % Проверка, что x не бесконечность
if ~isfinite(x), "Function func - input error", return,
end
y=x^2/2-x^4/4;
```

Теперь к функции func можно обращаться, как и к любой функции MATLAB, например к функции fplot для изображения графика или к функции fzero для отыскания корней нелинейного уравнения, см. главу 15 «Численный анализ в MATLAB»:

```
» fzero('func',1),
ans =
  1.4142e+000
```

Если используются глобальные переменные, то помимо описания в самой функции их следует описать также в командной строке или вызываемом script-файле. Например, создадим файл **massa.m**:

```
function out = massa(volume)
global DENSITY
out = DENSITY*volume;
```

Затем в командной строке наберем

```
» global DENSITY, DENSITY=1;
```

Обратимся к функции massa:

```
» massa(3)
ans =
     3
```

Глобальную переменную можно интерактивно менять в процессе работы, при этом нет необходимости модифицировать функцию massa. Для глобальных переменных разработчики MATLAB рекомендуют использовать заглавные буквы.

При помощи описателя persistent можно указать, что локальная переменная должна сохранять свое значение между вызовами функции. В этом случае переменная остается недоступной извне m-функции. При первом вызове такая переменная инициализируется пустым массивом.

Вычисление факториала может быть реализовано с применением описателя persistent следующим образом. Подготовим файл Fact_Pers.m:

```
function y=Fact_Pers(x)
persistent PERS
if isempty(PERS), PERS=1; end
y=PERS.*x; PERS=y;
```

Теперь можно найти 4!, организовав следующий цикл:

```
» for k=1:4, fa=Fact_Pers(k); end; fa
fa =
    24
```

Конечно, описанная функция плохо подходит для вычисления факториала. При повторном запуске цикла получим 576, так как значение переменной PERS сохраняется, *пока не завершен сеанс или не очищено рабочее пространство командой* clear.

Каждый m-файл может вызывать или ссылаться на другой m-файл, причем допустим рекурсивный вызов. Рекурсивной называется функция, которая вызывается из собственного тела. Приведем стандартный пример вычисления факториала. Для этого подготовим файл Fact_Rec.m следующего вида:

```
function y=Fact_Rec(x)
if x==1, y=1; return;
else y=x*Fact_Rec(x-1);
end
```

Обратимся к этой m-функции для вычисления факториала:

```
» Fact_Rec(4)
ans =
    24
```

По умолчанию доступны все m-файлы из текущего рабочего каталога, для указания других директорий имеется команда path. Пути можно задать также при помощи подсистемы Path Browser.

Система MATLAB содержит множество m-файлов, дополняющих базовый комплект. Эти файлы снабжены комментариями, из которых можно узнать их назначение, особенности применения, использованные алгоритмы и литературу. Напомним, что справочная система (команда help) позволяет получить перечень всех файлов и узнать назначение каждого. Ознакомиться с содержимым конкретного файла COMMAND можно, отыскав его в соответствующем каталоге либо просто набрав в строке ввода

```
type COMMAND
```

Типичный пример оформления файла дает функция rank для вычисления ранга матрицы. Файл rank.m располагается в подкаталоге \toolbox\matlab\matfun, и после выполнения команды

```
» type rank
```

на экране появится следующий текст:

```
function r = rank(A,tol)
%RANK   Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol=max(size(A))*norm(A)*eps.

%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.7 $  $Date: 1997/11/21 23:38:49 $

s = svd(A);
if nargin==1
   tol = max(size(A)') * max(s) * eps;
end
r = sum(s > tol);
```

В то же время при попытке вывести текст команды det получаем:

```
» type det
det is a built-in function.
```

Это означает, что команда det реализована в виде встроенной функции для повышения скорости ее выполнения.

В тексте команды rank использовано служебное слово nargin, означающее число поступивших в функцию параметров. Перечень зарезервированных терминов, обеспечивающих работу с функциями в случае переменного числа входных и выходных параметров, дан в табл. 12.9. Используя эти имена, можно создавать функции, способные обрабатывать обращения к функции с различным числом входных и выходных параметров. Например, из приведенного текста функции rank следует, что второй параметр — погрешность tol — является дополнительным. Если при обращении к функции rank его нет, то погрешность вычисляется по формуле tol=max(size(A)')*max(svd(A))* eps, где A — входная матрица, a eps — наименьшее вещественное число.

Таблица 12.9. Команды работы с параметрами функций

Команда	Назначение
nargin	Число входных параметров
nargout	Число выходных параметров
varargin	Список входных параметров
varargout	Список выходных параметров
nargchk	Сообщение о несоответствии числа входных параметров
inputname	Имя входного параметра

При выполнении функции может понадобиться вывести некоторую информацию, предупреждение или сообщение об ошибке. Для этого можно воспользоваться командами, приведенными в табл. 12.10.

Таблица 12.10. Команды вывода сообщений

Команда	Назначение
disp	Вывод сообщения на дисплей
warning	Предупреждающее сообщение
error	Сообщение об ошибке
lasterror	Сообщение о последней ошибке
errortrap	Игнорировать ошибки

Строки, передаваемые в качестве параметров, могут быть командами MATLAB, их интерпретация производится при помощи команды eval. Текстовая строка анализируется, и далее либо выводится сообщение об ошибке, либо воспринимается введенное выражение, оператор или команда. Например:

```
» s='2-cos(pi)'; eval(s)
ans =
     3
```

Если при наборе команды eval *сделана ошибка, то в командном окне появится сообщение о характере ошибки. Например:*

```
» eval("2cos(pi)")
??? 2
    |
Missing operator, comma, or semi-colon.
```

Вертикальная черта указывает на предполагаемое место ошибки, а текстовый комментарий перечисляет возможные причины. В аналогичной ситуации для команды eval, выполняющейся в m-функции, MATLAB выведет сообщение только о строке, в которой обнаружена ошибка. Для получения диагностики можно запустить команду eval, указав вторым параметром имя подготовленной m-функции с вызовом команды lasterror.

При численном интегрировании или решении дифференциальных уравнений требуется многократно вычислять значения функции, содержащей интегрируемое выражение или правую часть дифференциального уравнения. Для вычисления внешней функции применяется команда feval, имеющая следующий формат:

```
feval("FUN",PARAM)
```

По этой команде MATLAB передает параметры PARAM функции FUN и производит требуемое вычисление. Простой пример:

```
» feval("sin",[pi/2 0 pi/6])
ans =
    1.0000         0    0.5000
```

В MATLAB 6 допускается также обращение к функции с префиксом «@», тот же результат будет получен по команде

```
» feval(@sin,[pi/2 0 pi/6])
```

Для встроенных функций имеется аналогичная команда builtin.

Для проверки существования файла, переменной и других существует команда e=exist('a'), возвращающая целое число, по которому можно определить статус переменной или имени, см. табл. 12.11.

Таблица 12.11. Проверка существования объекта при помощи команды exist

Результат	Аргумент
7	Каталог
6	P-функция
5	Встроенная функция
4	Функция Simulink
3	mex-файл
2	m-файл
1	Переменная
0	Не относится ни к одному из перечисленных типов

В новых реализациях MATLAB появилась возможность введения подфункции subfunction. Для этого в один файл объединяются основная функция, дающая имя файлу, и следующие за ней несколько стандартно оформленных функций. Дополнительные функции могут вызываться только из основной функции. Для каждой функции из одного файла внутренние переменные являются локальными, а для обеспечения общего доступа к какой-нибудь переменной ее следует объявить глобальной. Например, файл interp1.m из каталога toolbox\matlab\polyfun содержит основную функцию interp1 и ряд подфункций: linear, cubic, nearest.

При отладке бывает необходимо подменить функцию, не удаляя ее из каталога. Для этого создается поддиректория с именем private, и тогда при совпадении имен функций первой будет обнаружена и соответственно станет использоваться функция из директории private. Это удобно для отладки собственных функций, которые вводятся, чтобы заменить стандартные тексты. При этом директорию private не нужно объявлять в списке имен path.

Работа с функциями происходит следующим образом. Когда MATLAB встречает новое имя, то сначала проверяет, не совпадает ли это имя с какой-нибудь переменной. Затем проверяется, не подфункция (subfunction) ли это. После этого просматривается private директория, и только потом имя функции ищется в списке path.

При первом обращении к функции MATLAB проводит лексический разбор текста и создает так называемый псевдокод, который хранится в памяти и используется в сеансе для работы. Благодаря этому экономится время расчета. Чтобы удалить из памяти псевдокод функции NAME, нужно использовать команду clear NAME. При помощи команды pcode можно сохранить псевдокод нужной функции, чтобы не проводить лексический анализ в последующих сеансах.

Отметим, что в MATLAB возможны два способа обращения к командам — обычный, когда имя команды command и список аргументов argument отделяются пробелами

```
command argument
```

а также вызов функции с тем же именем и параметрами, оформленными в виде строк:

```
command("argument")
```

Так, убрать оси на рисунке позволяют команды axis("off") и axis off (см. главу 14 «Графика MATLAB»). Использование функции удобнее в том случае, когда имена параметров argument формируются в процессе выполнения программы, например при выводе результатов расчета в файлы с именами, содержащими даты.

Функции inline

В MATLAB можно определить функцию одной переменной на сеанс при помощи команды inline. Например:

```
» fun=inline("x^2/2-x^4/4")
fun =
     Inline function:
     fun(x) = x^2/2-x^4/4
```

С введенной таким образом функцией можно выполнять различные операции. Продемонстрируем использование inline-функции для построения графика (команда fplot), нахождения корней (fzero) и вычисления интеграла (quad). Описания использованных команд даны в главах 14 «Графика MATLAB» и 15 «Численный анализ в MATLAB».

Вначале построим график:

```
» fplot(fun,[-1.5 1.5])
```

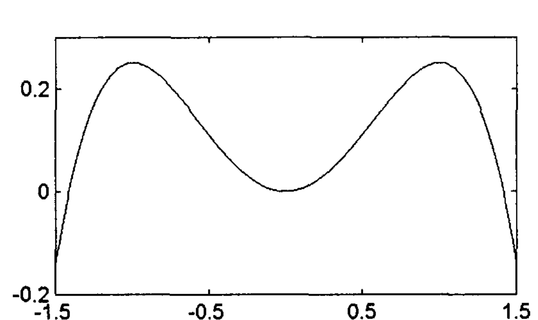


Рис. 12.2. Применение inline-функции для построения графика

Вычислим корень введенной функции, взяв в качестве начального приближения точку x=1:

```
» fzero(fun,1)
Zero found in the interval: [0.54745, 1.4525].
ans =
    1.4142
```

Нужно учитывать, что для ряда команд inline-функция должна быть определена в виде, допускающем векторизацию. Так, для введенной функции при вычислении интеграла появится сообщение об ошибке:

```
» quad(fun,0,1)
??? Error using ==> inline/feval
Error in inline expression ==> x^2/2-x^4/4
??? Error using ==> ^
Matrix must be square.
```

Поэтому функцию нужно переписать следующим образом:

```
» fun=inline("x.^2/2-x.^4/4");
```

Теперь интеграл будет вычислен:

```
» quad(fun,0,1)
ans =
    0.1167
```

При помощи inline-функции просто организовать решение дифференциального уравнения первого порядка.

Математические функции

Перечислим список элементарных математических функций MATLAB, применяемых к скалярным величинам (матрицы размера 1×1) или к каждому элементу матрицы-аргумента.

Таблица 12.12. Основные математические функции

Имя	Описание	Имя	Описание
abs	Абсолютная величина	exp	Экспонента
log	Натуральный логарифм	sign	Знак числа
log10	Десятичный логарифм	sqrt	Корень квадратный
real	Вещественная часть комплексного числа	imag	Мнимая часть комплексного числа
conj	Сопряженное число	angle	Фазовый угол
fix	Округление до ближайшего целого в сторону нуля	lcm	Наименьшее общее делимое
floor	Округление до ближайшего в сторону $-\infty$	ceil	Округление целого до ближайшего целого в сторону $+\infty$
gcd	Наибольший общий делитель	round	Округление до ближайшего по абсолютному значению целого числа
mod	Остаток от деления, вычисляемый по формуле mod(x,y)=x-y.*floor(x./y)	rem	Остаток от деления, вычисляемый по формуле rem(x,y)=x-y.*fix(x./y)

Перед вызовом любой функции полезно опробовать обращение к ней, пользуясь тем, что MATLAB интерпретатор. Например:

```
» sin([0 1; 2 pi])
```

```
ans =
        0    0.8415
   0.9093    0.0000
```

Реализация одной операции для всех элементов массива называется векторизацией.

Рассмотрим действие операций округления на примере нескольких чисел, оформленных в виде вектора-строки:

```
» fix([-2.5 -2.49 2.49 2.5])
ans =
    -2    -2     2     2
» floor([-2.5 -2.49 2.49 2.5])
ans =
    -3    -3     2     2
» ceil([-2.5 -2.49 2.49 2.5])
ans =
    -2    -2     3     3
» round([-2.5 -2.49 2.49 2.5])
ans =
    -3    -2     2     3
```

Таблица 12.13. Прямые тригонометрические и гиперболические функции

Функция	Тригономерическая функция	Гиперболическая функция
Косинус	cos	cosh
Синус	sin	sinh
Тангенс	tan	tanh
Котангенс	cot	coth
Секанс	sec	sech
Косеканс	csc	csch

Таблица 12.14. Обратные тригонометрические и гиперболические функции

Функция	Тригономерическая функция	Гиперболическая функция
Арккосинус	acos	acosh
Арксинус	asin	asinh
Арктангенс	atan	atanh
Аркотангенс	acot	acoth
Арксеканс	asec	asech
Арккосеканс	acsc	acsch

Список всех имеющихся математических функций может быть получен по команде

```
help elfun
```

Помимо обычного арктангенса atan имеется функция atan2, позволяющая корректно определить угол в радианах — арктангенс отношения ординаты и абсциссы.

Например:

```
» atan2(1,-sqrt(3))
ans =
    2.6180
```

Проверим ответ:

```
» pi*5/6
ans =
    2.6180
```

В заключение отметим, что рассмотрен базовый набор математических функций, специальные математические функции будут представлены в главе 15 «Численный анализ в MATLAB», а в главе 13 «Матричные вычисления» будут описаны функции от матриц.

13 ГЛАВА Матричные вычисления

Матрица – двумерный массив с вещественными или комплексными элементами – является основным объектом пакета MATLAB. Функции работы с матрицами разработаны так, чтобы обеспечить максимальную эффективность выполнения множества операций: от нахождения минимального элемента до сложных алгоритмов линейной алгебры. С подключением ядра пакета Maple в MATLAB стало возможным работать с матрицами, содержащими символьные переменные, аналитические выражения и другие элементы.

Наше изложение не предполагает детального описания всех команд работы с матрицами, а ориентировано на представление наиболее часто используемых команд. Полные перечни команд и перечисление всех вариантов обращения к ним следует искать в документации самой среды MATLAB и справочной литературе [11–21].

Операции над матрицами

Вначале рассмотрим функции MATLAB, предназначенные для обработки числовых массивов. При нахождении максимума и минимума, вычислении сумм, произведений, средних значений и т. д. для матриц по умолчанию действует следующее правило: вычисляется соответствующая операция для элементов столбцов и результат помещается в вектор-строку. Чтобы проделать вычисления построчно, можно вначале транспонировать исходную матрицу, а затем полученный в результате вектор-столбец превратить в строку. Кроме того, для многомерных массивов можно явно указать размерность, по которой будет действовать операция. Так, для массива `X` команды вычисления максимума и суммы по размерности `N` имеют соответственно следующий вид: `max(X,[],N)` и `sum(X,N)`.

Пусть задана следующая матрица второго порядка:

```
» A = [0 1 2; 3 4 6]
A =
     0     1     2
     3     4     6
```

Таблица 13.1. Команды работы с данными

Имя	Назначение
max	Определение максимальных элементов массива
min	Определение минимальных элементов массива
sum	Суммирование элементов массива
cumsum	Суммирование элементов массива с накоплением
prod	Произведение элементов массива
cumprod	Произведение элементов массива с накоплением
median	Определение медиан (срединных значений)
mean	Определение средних значений массива

Сравним результаты вычисления максимальных элементов в столбцах:

```
» max(A)
ans =
     3     4     6
```

и в строках:

```
» max(A')'
ans =
     2
     6
```

Другой способ вычисления максимальных элементов в строках имеет вид:

```
» max(A,[],2)
ans =
     2
     6
```

Нахождение наибольшего элемента исходной матрицы реализуется при помощи двукратного применения команды max:

```
» max(max(A))
ans =
     6
```

Имеются варианты команд max и min с иным числом входных и выходных параметров. Так, [Ma,In]=max(A) выведет вектор-строку максимальных значений Ma и их номера для каждого столбца In, а max(A,B) найдет результирующую матрицу, состоящую из максимальных значений двух матриц A и B.

Применение команды sum дает вектор-строку

```
» sum(A)
ans =
     3     5     8
```

Для пустых массивов MATLAB также выведет результат:

```
» sum([])
ans =
     0
```

```
» prod([])
ans =
     1
```

Команда cumsum позволяет провести суммирование, а cumprod — произведение элементов в режиме накопления, когда каждая последующая строка вычисляется как сумма или произведение предыдущих строк соответственно:

```
» cumprod(A)
ans =
     0     1     2
     0     4    12
```

Можно также явно указать размерность, по которой проводить суммирование. Так, накопительное суммирование по строкам обеспечивает операция

```
» cumsum(A,2)
ans =
     0     1     3
     3     7    13
```

Сравним результат вычисления средних значений по строкам:

```
» mean(A')'
ans =
    1.0000
    4.3333
```

и операцию вычисления медиан (срединных значений) по строкам:

```
» median(A')'
ans =
     1
     4
```

Команды mean и median применяются для анализа данных, см. главу 15 «Численный анализ в MATLAB», а список функций анализа данных можно получить, вызвав справку

```
help datafun
```

Таблица 13.2. Команды сортировки

Имя	Назначение
sort(X,N)	Сортировка по возрастанию (упорядочивание по модулю) элементов массива X по размерности N
sortrows(X,N)	Сортировка строк с упорядочиванием по элементам столбца с номером N
cplxpair(X,TOL,N)	Сортировка комплексно-сопряженных пар (упорядочивание по вещественной части) с точностью TOL по размерности N

Сортировку элементов массива по строкам делает команда sort. Если массив составлен из комплексных чисел, то сортировка ведется для абсолютных значений (модулей чисел), В случае их равенства сначала будет выведено число с отрицательной мнимой частью. Например:

```
» sort([5 1+5*i 6 1-5*i])
```

```
ans =
   5.0000      1.0000 - 5.0000i   1.0000 + 5.0000i   6.0000
```

Приведем пример сортировки комплексных пар при помощи команды cplxpair, которая расставляет пары по величине вещественной части, начиная с меньших значений. В случае равенства вещественной части, сначала будет выведено число с отрицательной мнимой частью. Затем команда выводит вещественные числа:

```
» cplxpair([-i 4+i 4-i 2 i])
ans =
  Columns 1 through 4
   0-1.0000i     0+1.0000i   4.0000-1.0000i   4.0000+1.0000i
  Column 5
   2.0000
```

Здесь вывод результата потребовал нескольких строк.

Сортировку матрицы по строкам поясняет следующий пример. Зададим матрицу из трех строк:

```
>> B=[0 3 2; 4 2 2; 1 3 2]
B =
     0     3     2
     4     2     2
     1     3     2
```

Применим команду сортировки строк (по умолчанию сортировка ведется по элементам первого столбца):

```
>> sortrows(B)
ans =
     0     3     2
     1     3     2
     4     2     2
```

Теперь укажем, что сортировку следует вести по элементам третьего столбца, а при совпадении элементов использовать второй столбец:

```
>> sortrows(B,[3 2])
ans =
     4     2     2
     0     3     2
     1     3     2
```

Таблица 13.3. Функции от матриц

Имя	Описание
expm	Матричная экспонента
sqrtm	Квадратный корень из матрицы
logm	Логарифм от матрицы
funm	Функция от матрицы

Достаточно легко найти целую положительную степень матрицы и вычислить полином от матрицы. Если матрица A квадратная и невырожденная, то вычисле-

ние A^(-3) состоит в нахождении обратной матрицы и возведения ее в куб. Дробные степени вычисляются также, а результат зависит от распределения собственных чисел матрицы. Специальные команды, реализующие важные для приложений операции вычисления экспоненты, логарифма и функции общего вида, даны в табл. 13.3.

Команда exp(A) находит экспоненту для каждого элемента матрицы A, а для определения матричной экспоненты e^A имеются команда expm(A), основанная на разложении Паде, и функция expm2(A), находящая экспоненту с помощью разложения Тейлора. Вычисление любой матричной функции можно реализовать при помощи функции funm. Разработчики MATLAB предупреждают, что функция от матрицы вычисляется с использованием потенциально неустойчивого алгоритма, поэтому результаты могут быть не точны. Представим простой пример вычисления косинуса от матрицы второго порядка при помощи функции funm и сравним результат с вычислением на основе матричной компоненты:

```
» A=[2 1; -1 4], S=funm(A,'cos'), real(expm(i*A))
A =
     2     1
    -1     4
WARNING: Result from FUNM may be inaccurate. esterr = 1
S =
   -0.9900         0
         0   -0.9900
ans =
   -0.8489   -0.1411
    0.1411   -1.1311
```

Заметим, что использование funm привело к предупреждению и плохому результату. Если даже незначительно изменить первый элемент матрицы A(1,1), то получится правильный результат.

```
» A=[2.1 1; -1 4]; S=funm(A,'cos'), real(expm(i*A))
S =
   -0.9564   -0.0930
    0.0930   -1.1331
ans =
   -0.9564   -0.0930
    0.0930   -1.1331
```

Линейная алгебра

Теперь рассмотрим команды, реализующие ряд важных операций линейной алгебры (см. табл. 13.4).

Вычисление p-нормы матрицы или вектора M производится при помощи команды

```
norm(M,p)
```

Для вектора p-норма вычисляется при помощи формулы

```
sum(abs(M).^p)^1/p
```

Таблица 13.4. Матричные характеристики

Имя	Описание
det	Вычисление определителя
trace	Вычисление следа матрицы
cond	Вычисление числа обусловленности
condest	Оценка числа обусловленности
rcond	Параметр обусловленности
rank	Определение ранга матрицы
norm	Вычисление нормы матрицы (вектора)
normest	Оценка нормы матрицы (вектора)

Приведем пример вычисления различных норм. Подготовим вектор-строку v и набор значений параметра p, для которых будем вычислять нормы вектора v:

```
» v=1:3, p=[-inf,1,2,3,inf]
v =
     1     2     3
p =
  -Inf     1     2     3   Inf
```

Запустим вычисления при помощи цикла for ... end:

```
» for k=p, [k norm(v,k)], end
```

В результате получим таблицу, где левая колонка дает порядок нормы, а правая — вычисленное значение (строки с переменной ans опущены):

```
  -Inf     1
     1     6
    2.0000    3.7417
    3.0000    3.3019
   Inf     3
```

Отметим, что здесь для представления результата использован формат short, и если выводимая строка не содержит вещественных чисел, то мантисса опускается.

Обращение к функции norm без второго параметра эквивалентно вызову norm(v,2). Нормы матриц вычисляются для значений p, равных 1, 2, Inf, и так называемой нормы Фробениуса: norm(A,'fro')=sqrt(sum(diag(A'*A))).

Число обусловленности характеризует точность операции обращения матрицы или решения системы линейных уравнений и равно отношению максимального сингулярного числа к минимальному. Функция rcond(A) вычисляет значение, практически обратное числу обусловленности матрицы A. Это число близко к единице, если матрица хорошо обусловлена, и к нулю — если матрица плохо обусловлена.

Определитель матрицы A вычисляется методом Гаусса на основе треугольного разложения.

Продемонстрируем примеры обращения к ряду функций таблицы. Вычислим для гильбертовых матриц четных порядков их ранг, различные нормы, а также десятичные логарифмы числа обусловленности и определителя:

```
» for k=3:3:12,
H=hilb(k);
[k, rank(H), norm(H,'fro'), norm(H,2), norm(H,Inf), ...
log10(cond(H)), log10(det(H))]
end
```

Далее приведены результаты вычислений, причем для экономии места опущены строки присвоения ответа переменной ans, а значения порядка матрицы и ранга представлены целыми числами:

```
3    3    1.4136    1.4083    1.8333    2.7194   -3.3345
6    6    1.6370    1.6189    2.4500    7.1747   -17.2702
9    9    1.7559    1.7259    2.8290   11.6930   -42.0123
12   11    1.8358    1.7954    3.1032   16.2464  -77.5439
```

Таблица 13.5. Команды линейной алгебры

Имя	**Описание**
inv	Вычисление обратной матрицы
pinv	Вычисление псевдообратной матрицы
null	Определение ядра (нуль-пространства) матрицы
orth	Вычисление ортонормального базиса

Матрица с разным числом строк и столбцов не имеет обратной и определителя. Частичное решение проблемы обращения такой матрицы представляет псевдообратная матрица Мура–Пенроуза, которая находится при помощи команды pinv. Приведем пример:

```
» A=[1 2 3; 2 1 3]
A =
     1     2     3
     2     1     3
» B=pinv(A)
B =
   -0.4444    0.5556
    0.5556   -0.4444
    0.1111    0.1111
```

Проверим, что произведение A*B дает единичную матрицу размера 2×2:

```
» C=A*B
C =
    1.0000   -0.0000
   -0.0000    1.0000
```

Другой порядок произведения дает матрицу размера 3×3 с практически нулевым определителем:

```
» D=B*A; det(D)
ans =
  -5.5511e-017
```

Решение систем линейных уравнений

В различных приложениях требуется находить решения систем линейных алгебраических уравнений. Пусть дана система AX=B с известными матрицей A и вектором (или матрицей) B. Ее решение относительно неизвестного вектора (или матрицы) X представляется формулой $X=A^{-1}B$. В MATLAB это записывается как X=A\B. Если же система записана в виде XA=B, то ее решением будет X=B/A. Размерности массивов должны быть согласованы: необходимо, чтобы значение size(A,2) было равно size(B,1), и тогда матрица-результат будет иметь размерность size(A,1)×size(B,2). Для квадратных матриц ищется точное решение, для переопределенных систем size(A,1)>size(A,2) используется метод наименьших квадратов, а для недоопределенных систем ищется решение с числом size(A,1) ненулевых компонент.

Приведем простые примеры решения систем малой размерности. Начнем с квадратной матрицы:

```
» A=[0 1; 2 1], B=[1 3]', X=A\B
A =
     0     1
     2     1
B =
     1
     3
X =
     1
     1
```

Проиллюстрируем правое деление (решается система XA=B1):

```
» B1=[1 2]; B1/A
ans =
    1.5000    0.5000
```

Для недоопределенной системы (число уравнений меньше числа неизвестных) получаем:

```
» A(1,3)=2
A =
     0     1     2
     2     1     0

» A\B
ans =
    1.5000
         0
    0.5000
```

Переопределим матрицу, чтобы число строк превысило число столбцов:

```
» A(:,3)=[];A(3,:)=[3 2]
A =
     0     1
     2     1
     3     2
```

Добавим третий элемент в вектор B:

```
» B(3)=1; B'
ans =
     1     3     1
```

Теперь для переопределенной системы AX=B (три уравнения и две неизвестных компоненты вектора X) имеем:

```
» X=A\B
X =
    0.4286
    0.4286
```

Некоторые алгоритмы решения систем линейных алгебраических уравнений основаны на факторизации матриц по одной из схем: Холецкого, метода Гаусса и QR-разложения. Например, факторизация при помощи команды chol позволяет переписать систему Ax=b в виде R'Rx=b, и тогда решение дается формулой x=R\(R'\b). В случае матрицы порядка n для получения разложения необходимо число операций порядка n^3, а для определения решения — порядка n^2. Если решаются системы с одной матрицей и различными правыми частями, то для ускорения вычислений лучше один раз провести факторизацию матрицы и вычислить обратную матрицу.

Таблица 13.6. Команды факторизации

Имя	Описание
chol	Разложение Холецкого для симметричных, положительно определенных матриц
cholnc	Неполное разложение Холецкого (представление симметричной матрицы в виде произведения верхней треугольной и транспонированной матриц)
lu	LU-разложение (для квадратных матриц)
luinc	Неполное LU-разложение
hess	Приведение к форме Хессенберга
rref	Приведение к треугольной форме
qr	QR-разложение (представление матрицы в виде произведения ортогональной и верхней треугольной матриц)

Испытаем некоторые функции из таблицы, взяв в качестве аргумента матрицу Паскаля третьего порядка:

```
» A=pascal(3)
A =
     1     1     1
     1     2     3
     1     3     6
```

Разложение Холецкого записывается следующим образом:

```
» chol(A)
ans =
     1     1     1
     0     1     2
     0     0     1
```

Разбиение матрицы на произведение нижней (L) и верхней (U) треугольных матриц, а также матрицы перестановок (P) делается одной командой:

```
>> [L,U,P]=lu(A)
L =
    1.0000         0         0
    1.0000    1.0000         0
    1.0000    0.5000    1.0000
U =
    1.0000    1.0000    1.0000
         0    2.0000    5.0000
         0         0   -0.5000
P =
     1     0     0
     0     0     1
     0     1     0
```

При обращении [L,U]=lu(A) матрица L есть результат перемножения нижней треугольной матрицы и матрицы перестановок. Для системы Ax=b метод LU-разложения дает решение в виде x=U\(L\b).

Функция qr (QR-разложение) дает факторизацию, в результате которой возникают верхне-треугольная матрица R и матрица Q. Для исходной вещественной матрицы получается матрица Q с ортонормированными столбцами, а для матрицы с комплексными коэффициентами Q — еще и унитарная матрица. Например, для матрицы Паскаля третьего порядка получим:

```
» [Q,R]=qr(A)
Q =
   -0.5774    0.7071    0.4082
   -0.5774         0   -0.8165
   -0.5774   -0.7071    0.4082
R =
   -1.7321   -3.4641   -5.7735
         0   -1.4142   -3.5355
         0         0    0.4082
```

Конечно, рассмотренные примеры элементарны. Для сложных задач могут возникнуть проблемы с отысканием решения. В этих случаях возможно появление следующих сообщений.

Пусть n — порядок матрицы. Если в течение 30n итераций решение не получено, то выводится сообщение о том, что решение не сходится:

```
Solution will not converge
```

При решении системы могут быть выведены следующие диагностические сообщения:

- Matrix is singular to working precision — матрица вырождена для выбранной точности;
- Divide by zero — деление на нуль при поэлементном делении;
- Warning: Matrix is closed to singular or badly scaled — предупреждение: матрица близка к вырожденной или плохо масштабирована (при этом будет выведено число обусловленности);

- Warning: rank deficient, rank = xxx tol = xxx — предупреждение: неполный ранг, ранг = xxx точность = xxx.

Спектр и сингулярное разложение

Обобщенная проблема собственных значений заключается в нахождении нетривиальных решений системы уравнений Ax=λBx, здесь A, B — квадратные матрицы порядка n, x — обобщенный собственный вектор (вектор-столбец), λ — обобщенное собственное значение (число). Для нахождения собственных чисел используется ряд модулей пакета EISPACK:

- операции масштабирования и восстановления производят balance и balbak;
- приведение матрицы к форме Хессенберга осуществляет команда orthes;
- запоминание преобразований производится командой ortan;
- hqr2 вычисляет собственные значения и векторы матрицы, приведенной к верхней форме Хессенберга.

Решение обобщенной проблемы собственных значений использует модули EISPACK, основанные на QZ-алгоритме: qzhes, qzit, qzval, qzvec.

Таблица 13.7. Команды вычисления спектра

Имя	Описание
poly	Вычисление характеристического полинома
polyeig	Вычисление собственных значений матричного полинома
eig	Вычисление собственных чисел и векторов
schur	Декомпозиция (разложение) Шура
svd	Сингулярное разложение матрицы (SVD-разложение)

Для определения собственных значений и собственных векторов матрицы A служит команда

```
[U,D]=eig(A)
```

Здесь диагональная матрица D состоит из собственных чисел, а матрица U составлена из собственных векторов-столбцов матрицы A. Если в левой части указан единственный выходной параметр, то результатом будет выступать вектор-столбец собственных чисел eig(A).

Для решения обобщенной задачи A=λB используется обращение

```
[U,D]=eig(A,B)
```

Например, определим собственные числа и векторы для матрицы Паскаля третьего порядка:

```
» P=pascal(3);
» [U,D]=eig(P)
U =
    0.5438   -0.8165    0.1938
```

```
   -0.7812   -0.4082    0.4722
    0.3065    0.4082    0.8599
D =
    0.1270         0         0
         0    1.0000         0
         0         0    7.8730
```

Теперь проверим, что первый столбец матрицы U является собственным вектором матрицы A, отвечающим первому собственному числу:

```
» k=1; P*U(:,k)-D(k,k)*U(:,k)
ans =
  1.0e-015 *
    0.2220
    0.7772
    0.9784
```

Собственное число и отвечающий ему собственный вектор найдены с точностью до шестнадцатого знака. Проверку можно было провести и для всех векторов сразу P*U-U*D.

Превратить диагональную матрицу собственных значений в вектор поможет команда diag:

```
» diag(D)'
ans =
    0.1270    1.0000    7.8730
```

Для решения проблемы собственных значений может быть использована декомпозиция Шура:

```
[U,T]=schur(A)
```

Если матрица A действительная, то выводится матрица T с действительными собственными числами на диагонали, а комплексные числа представляются в виде блоков 2×2 (почти треугольная форма). Для комплексной матрицы получается комплексная форма Шура в виде верхней треугольной матрицы с собственными значениями на диагонали. Матрица U является унитарной матрицей: произведение U*U' есть единичная матрица. Обращение T=schur(A) выводит только матрицу в форме Шура. Для преобразования полученной матрицы из действительной формы в комплексную применяются функция [U,T]=rsf2csf(U,T); функция [U,T]=csf2rsf(U,T) дает обратное преобразование.

Для решения матричных уравнений Сильвестра и ряда других задач линейной алгебры нужно, чтобы к форме Шура приводилась пара матриц. Для этого имеется команда qz. Положенный в основу этой функции алгоритм был реализован первоначально в рамках пакета EISPACK, многие процедуры которого использованы в системе MATLAB.

После включения в MATLAB ядра пакета Maple можно использовать функции из Maple. Покажем, как с помощью функции jordan найти жорданову структуру матрицы с кратными собственными числами. Введем матрицу A:

```
» A=[2 -6 0; 2 -4 2; 0 1 1];
```

Каноническая жорданова форма матрицы получится по команде:

```
» [B,J]=jordan(A)
B =
    -6     9    -8
    -2     4    -4
     2    -2     2
J =
     0     1     0
     0     0     0
     0     0    -1
```

На диагонали матрицы J находятся собственные числа и двукратному нулевому собственному значению отвечает жорданова клетка. Прежде чем воспользоваться следующей командой, изменим формат представления чисел:

```
» format short e
```

Теперь для той же матрицы A найдем разложение Шура:

```
 » [U,S]=schur(A)
U =
  8.7287e-001 -2.6726e-001 -4.0825e-001
  4.3644e-001  8.0178e-001  4.0825e-001
 -2.1822e-001  5.3452e-001 -8.1650e-001
S =
 -1.0000e+000 -6.1237e+000 -4.5434e+000
            0 -1.6741e-008 -2.6186e+000
            0 -2.0912e-021  1.6741e-008
```

Здесь нулевое кратное собственное значение должно было бы определиться из нижнего блока матрицы S. Вследствие погрешности вычислений получаем маленькие ненулевые собственные значения:

```
» eig(S)'
ans =
 -1.0000e+000 -1.6741e-008  1.6741e-008
```

Если обратиться к команде eig, то вычисление даст следующие собственные числа:

```
» eig(A)
ans =
 -1.0000e+000
  7.7802e-016 +3.5798e-008i
  7.7802e-016 -3.5798e-008i
```

Функция svd определяет сингулярное разложение матрицы. Сингулярное число σ и соответствующие ему векторы u и v матрицы A удовлетворяют равенствам Av=σu, A'u=σv. Здесь A' — транспонированная матрица, σ — вещественное число. Образуем матрицу S, в которой расположим на диагонали сингулярные числа. Тогда AV=US, A'U=VS и A=USV'. Диагональ матрицы S состоит из положительных значений квадратных корней матрицы A'A. Если матрица A симметричная и положительно определенная, то сингулярные числа совпадают с собственными числами матрицы A. Вектор сингулярных чисел получается при обращении с одним выходным параметром S=svd(A). Приведем пример вычисления сингулярных чисел для прямоугольной матрицы:

```
» A=[ 2 3 4; 7 5 3]
A =
     2     3     4
     7     5     3
» [U,S,V]=svd(A)
U =
    0.4743    0.8803
    0.8803   -0.4743
S =
   10.2514         0         0
         0    2.6284         0
V =
    0.6937   -0.5934    0.4082
    0.5682    0.1025   -0.8165
    0.4427    0.7983    0.4082
```

Перемножая полученные матрицы, приходим к исходному массиву:

```
» U*S*V'
ans =
    2.0000    3.0000    4.0000
    7.0000    5.0000    3.0000
```

Значения сингулярных чисел используются при определении ряда характеристик матриц. При обращении rank(A) ранг матрицы A определяется как количество сингулярных чисел, превышающих порог max(size*A))*norm(A)*eps, а в случае команды rank(A,tol) порог дается величиной tol.

Работа с разреженными матрицами

По умолчанию MATLAB проводит матричные вычисления, оперируя со всеми элементами матриц. Если число элементов велико, то выполнение операций требует большого времени. Для многих задач характерно наличие матриц определенной структуры, с большим количеством нулевых элементов. Поэтому в MATLAB имеются две системы хранения матриц: полная и разреженная. В первом варианте нулевые элементы хранятся в памяти, и для хранения матрицы с N строками и M столбцами требуется место для N*M чисел. Для разреженных матриц хранятся только ненулевые элементы. Запись осуществляется по столбцам, а номера строк хранятся в целочисленном массиве.

Ввод разреженной матрицы наиболее просто реализуется с помощью команды sparse:

```
S = sparse(m,n,V)
```

Здесь буквы m и n обозначают индексы строки и столбца, куда будет введен вектор V, причем в качестве индекса можно использовать диапазон. Нулевые элементы при этом автоматически удаляются. Например:

```
» S = sparse(3:5, 3, [1 0 2])
S =
   (3,3)        1
   (5,3)        2
```

Команда sparse(V) преобразует полную матрицу V в разреженную. С разреженными матрицами можно производить те же операции, что и с обычными, необходимое для этого время будет пропорционально количеству арифметических операций над ненулевыми элементами. Применение некоторой функции только к ненулевым элементам поддерживает команда spfun — это аналог функции feval (см. главу 16 «Программирование в MATLAB»).

Ряд команд предоставляет стандартные возможности по вводу матриц специального вида. Например, для ввода единичной разреженной матрицы имеется команда speye, а случайная матрица создастся по команде sprandn. Симметричная матрица со случайным заданием элементов будет выведена по команде sprandnsym.

Аналогом команды diag для работы с разреженными матрицами является команда spdiags, которая позволяет выделить указанные диагонали, заменить их, выделить все ненулевые диагонали, сформировать разреженную матрицу по заданным диагоналям.

Найти индексы ненулевых элементов разреженной матрицы позволяет команда find.

Чтобы определить такие характеристики разреженной матрицы, как норма, число обусловленности и ранг матрицы применяются, соответственно, команды normest, condest и sprank.

Разреженную матрицу в нормальную можно преобразовать командой full. Чтобы проверить матрицу на принадлежность к категории разреженных, служит команда issparse.

Количество ненулевых элементов можно определить командой nnz, а сформировать вектор из ненулевых элементов — функцией nonzeros.

Имеются варианты команд определения спектральных характеристик (eigs) и сингулярных чисел (svds), позволяющие вычислять не весь набор величин, а некоторое их количество. Например, команда eigs(A,K,VAL) вычислит K собственных чисел матрицы A, расположенных около значения VAL. Эти команды позволяют избежать огромных затрат времени на решение полных проблем. Отметим, что данные команды действуют и для обычных матриц, но особенно удобны для анализа больших разреженных матриц.

Команды load и save поддерживают работу с разреженными матрицами.

Если матрица задается при помощи формата ASCII, то конвертировать данные из подготовленного внешнего файла поможет команда spconvert, при этом ненулевые элементы задаются построчно: на отдельной строке должны быть индексы и значение элемента.

Целая серия команд реализует алгоритмы упорядочения матриц, и результатом действия этих команд является вектор упорядоченности. Упорядочение автоматически применяется MATLAB при решении линейных систем с разреженными матрицами и при операциях обращения. Команда symrcm находит такой вектор упорядочения симметричной матрицы, чтобы ненулевые элементы были сгруппированы вблизи главной диагонали. Команды symmmd и colmmd позволяют получить более разреженное, чем исходная матрица, LU-разложение для симметричной и несимметричной матрицы соответственно.

В задачах, полученных в результате применения метода конечных элементов, имеется информация о пространственных координатах, связанных со строками и столбцами разреженной матрицы. Пусть в матрице XY содержатся координаты конечноэлементной сетки, а S — соответствующая матрица связности. Нарисовать соответствующий граф можно при помощи команды

```
gplot(S,XY)
```

Картина распределения ненулевых элементов будет выведена по команде

```
spy(S)
```

Приведем пример преобразования матрицы Пуассона (из стандартного набора матриц gallery) при сетке 6×6 в разреженную и последующие выводы структур матриц, полученных при помощи различных преобразований:

```
» Poi=gallery("poisson",6); S=sparse(Poi);r=symrcm(S);
```

Для размещения на одном рисунке нескольких изображений используется команда subplot (см. главу 14 «Графика MATLAB»).

```
» subplot(2,2,1),spy(Poi),subplot(2,2,2),spy(S(:,r))
subplot(2,2,3),spy(lu(S)),subplot(2,2,4),spy(lu(S(:,r)))
```

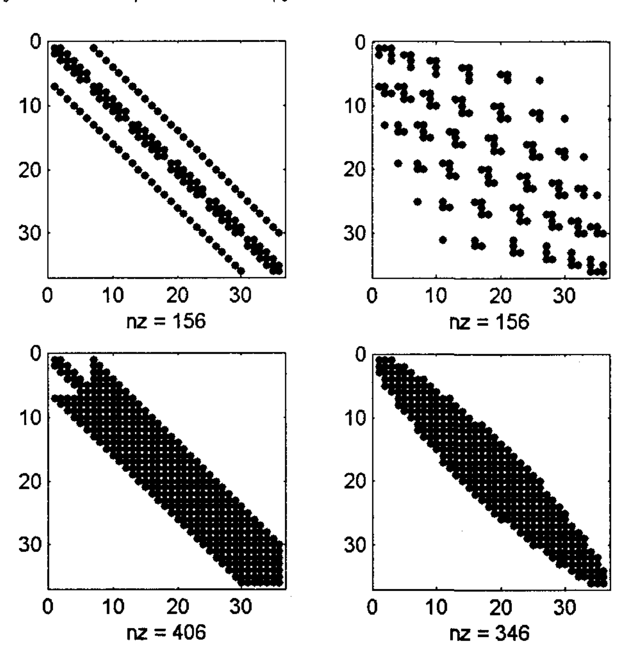


Рис. 13.1. Исходная задача для задачи Пуассона и преобразованные матрицы

Внизу каждой картины дано число ненулевых элементов для матрицы размера 36×36.

Решение системы линейных алгебраических уравнений с разреженной матрицей может быть получено прямыми методами (различные варианты метода Гаусса — LU-факторизация, схема Холецкого) и итерационными методами. Прямые методы быстрее и лучше, если достаточно памяти для проведения соответствующих

преобразований. Эти методы располагаются в ядре MATLAB, и их реализация осуществлена с большой тщательностью. Итерационные методы выручают, если матрица системы большого размера, как, например, при решении уравнений в частных производных сеточным методом или методом конечных элементов. Наиболее распространены методы bicg, bicstab, cgs, gmres, gmres, gmr. Все шесть методов можно предварять проведением левого или правого предобусловливания, позволяющего улучшить обусловленность системы перед применением итерационного метода. Полный перечень команд работы с разреженными матрицами может быть получен по команде справки

```
help sparfun
```

14 ГЛАВА Графика MATLAB

Для многих исследований и вычислительных работ важным этапом является визуализация данных, поддержка которой осуществляется в MATLAB при помощи набора мощных графических команд высокого уровня и программируемого интерфейса дескрипторной графики Handle Graphics.

Команды высокоуровневой графики реализуют построение графиков кривых в двумерном и трехмерном пространствах, изображение поверхностей, рисование линий уровня, гистограмм, многие виды специализированной графики и анимацию. При этом управление цветом, масштабирование, нанесение подписей, маркировка осей и т. д. осуществляется достаточно просто, а назначенные по умолчанию режимы вполне удовлетворительны в большинстве случаев. Начиная с версии 5.3 под Windows появилась возможность интерактивного оформления рисунков. Объектно-ориентированная система Handle Graphics предоставляет доступ ко всем характеристикам графических объектов и позволяет создавать новые графические команды.

Кривые, поверхности и другие графические объекты строятся в специальном графическом окне (figure). Первое обращение в сеансе к графической команде автоматически вызывает появление окна, которому присваивается первый номер. Чтобы организовать новое окно или перейти от одного окна к другому, нужно выбрать пункт New Figure в меню File командного или графического окна. Для того чтобы перейти к имеющемуся окну с номером N или организовать новое окно, достаточно в строке ввода ввести команду

```
figure N
```

Одновременно может быть открыто несколько графических окон. В многооконной среде действуют обычные способы перемещения от приложения к приложению: выбор при помощи панели задач или нажатие Alt+Tab. При описании графических команд будем приводить рисунки без меню и значков, то есть в том виде, который нужен исследователю для вставки графиков и рисунков в отчеты, статьи, книги, а в конце главы рассмотрим интерфейс графического окна и имеющиеся в нем возможности для редактирования изображений и оформления рисунков. В MATLAB для сохранения рисунка имеется большой выбор форматов графиче-

ских файлов, причем рисунок можно записать, используя меню графического окна или выполнив команду в основном окне.

Двумерная графика

Основная команда двумерной графики plot строит графики кривых с абсциссами X1, X2, ... и ординатами Y1, Y2, ... и имеет следующий формат:

```
plot(X1,Y1,S1,X2,Y2,S2,...)
```

Дополнительные параметры, представленные строками S1, S2, ... позволяют задать тип, толщину и цвет рисуемой кривой, а также форму и размер маркера. Сводка символов, используемых для определения типа и цвета линий, а также типа меток, представлена в табл. 14.1–14.3. Строки S1, S2, ... могут быть подготовлены отдельно либо информация по оформлению кривой должна быть взята в апострофы. Если дополнительные параметры опущены, то MATLAB устанавливает параметры, действующие по умолчанию: все кривые будут выведены сплошными линиями и окрашены циклически в шесть различных цветов. Если ординаты Y1, Y2, ... представлены двумерными массивами, то строятся графики для столбцов. Если массивы абсцисс и ординат содержат комплексные элементы, то их мнимые части игнорируются и график строится по вещественным частям.

Таблица 14.1. Типы линий

Символ	Описание	Символ	Описание
-	сплошная	--	пунктирная
:	пунктирная	-.	штрихпунктирная

Таблица 14.2. Типы меток

Символ	Описание	Символ	Описание
.	точка	+	плюс
*	звезда	x	крест
o	кружок	d	ромб
s	квадрат	^ v > <	различные треугольники
P pentagram	пятиконечная звезда	H hexagram	шестиконечная звезда

Таблица 14.3. Цвет линий

Символ	Цвет	Код RGB
y	желтый	(1 1 0)
m	фиолетовый	(1 0 1)

продолжение ⇗

Таблица 14.3 (*продолжение*)

Символ	Цвет	Код RGB
c	голубой	(0 1 1)
r	красный	(1 0 0)
g	зеленый	(0 1 0)
b	синий	(0 0 1)
k	черный	(0 0 0)
w	белый	(1 1 1)

Размерности массивов, задающих абсциссы и ординаты, должны совпадать. В противном случае будет выведено сообщение об ошибке:

```
» x=1:3; y=1:4; plot(x,y)
??? Error using ==> plot
Vectors must be the same lengths.
```

Начнем с построения графика единственной кривой, указав в апострофах, что рисуется сплошная кривая черного цвета, см. рис. 14.1:

```
» x=0:.01:3; y=exp(x).*sin(9*x); plot(x,y,'-k')
```

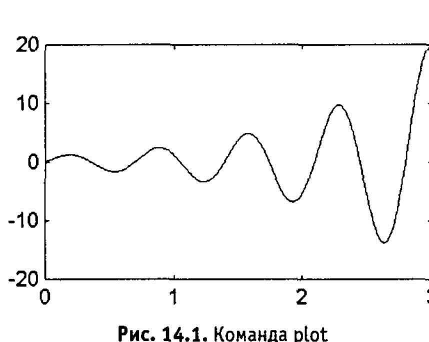


Рис. 14.1. Команда plot

Здесь вектор x содержит узлы сетки на отрезке [0,3], вычисленные с шагом 0.01, а вектором y даются значения функции (экспонента на синус) в этих узлах.

Назначения цветов, маркеров и типов линий для кривых можно изменить после рисования, используя меню графического окна, см. далее подраздел этой главы «Интерактивная работа с графикой».

Полный доступ к свойствам рисунка дается при помощи его дескриптора. Для этого результат выполнения графической команды, создающей рисунок, нужно присвоить какой-нибудь переменной — дескриптору, и затем использовать эту переменную для чтения и изменения полей специальной структуры, описывающей все свойства построенного рисунка. Смотрите об этом подробнее в подразделе «Элементы дескрипторной графики».

Например, график той же функции можно вывести пунктирной кривой черного цвета с маркерами в виде кружка размера 10, а результат присвоить переменной p. (Сам рисунок не приводится.)

```
» p=plot(x,y,'k:o','MarkerSize',10)
p =
   74.0045
```

Информацию о рисунке можно выводить по команде get. В приводимом листинге опущена часть информации о созданном ранее рисунке:

```
» get(p)
                    Color = [0 0 0]
                    EraseMode = normal
                    LineStyle = :
                    LineWidth = [0.5]
                    Marker = o
                    MarkerSize = [10]
                    MarkerEdgeColor = auto
                    MarkerFaceColor = none
                    XData = [ (1 by 301) double array]
                    YData = [ (1 by 301) double array]
                    ZData = []
```

Чтобы нарисовать на одном графике несколько кривых, нужно попарно перечислить векторы, задающие абсциссы и ординаты. Например:

```
» x=-2:.1:2; y=cos(2*x); z=sin(x);
plot(x,y,'k',y,z,'k:.'), legend("cos(2x)",'z vs y')
```

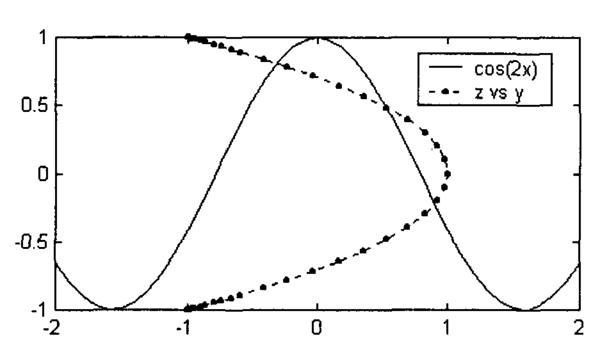


Рис. 14.2. Рисование нескольких кривых на одном графике

Отметим, что для первой пары (x,y) вектор y задает ординаты, а для второй пары (y,z) вектор y дает абсциссы. Указанный в апострофах символ k задает черный цвет для обеих кривых; двоеточие означает, что вторая кривая будет нарисована пунктиром, а точка — что в качестве маркера на второй кривой будет использована точка. Кроме того, для идентификации кривых применена команда legend, позволяющая связать с каждой кривой некоторую текстовую информацию. Вторая пара векторов дает также пример параметрического задания графика.

Другой способ размещения на одном графике несколько кривых состоит в том, чтобы сформировать матрицу, столбцы которой будут содержать нужные ординаты:

```
» x=0:.02:pi; Y=[sin(2*x)',sin(3*x)',sin(4*x)']; plot(x,Y)
```

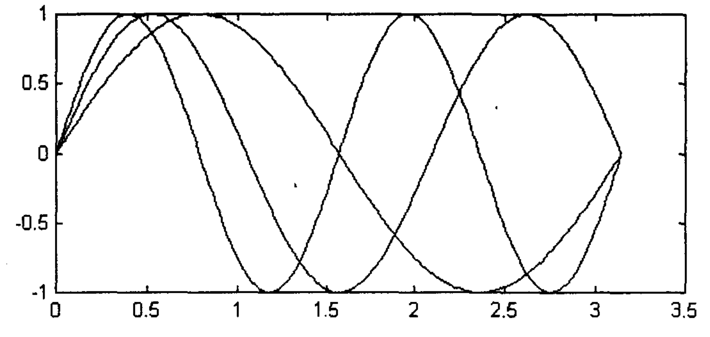


Рис. 14.3. Построение нескольких графиков при помощи двумерного массива ординат

Здесь апостроф использован, чтобы перевести векторы ординат, заданные строками, в столбцы. Вектор, определяющий абсциссы, может быть строкой или столбцом, и тот же результат получится, если использовать команду `plot(x',Y)`. Отметим, что MATLAB по умолчанию старается маркировать оси, используя целые числа или их доли, так что в качестве правой границы выбрано значение 3.5, а не π, как определено массивом `x`.

Для рисования графиков в логарифмическом масштабе имеются следующие команды: `loglog` — логарифмический масштаб по обеим осям, `semilogx (semilogy)` — логарифмический масштаб по координате `x (y)`.

График функции в MATLAB можно нарисовать с помощью команды `fplot`. Введем функцию `sinx`^{2}, создав соответствующий m-файл **sinx2.m**:

```
function y=sinx2(x)
y=sin(x.^2)
```

Здесь для вычисления квадрата аргумента использовано поэлементное возведение в степень (`.^`). Теперь график функции можно вывести по команде

```
» fplot("sinx2",[0,pi])
```

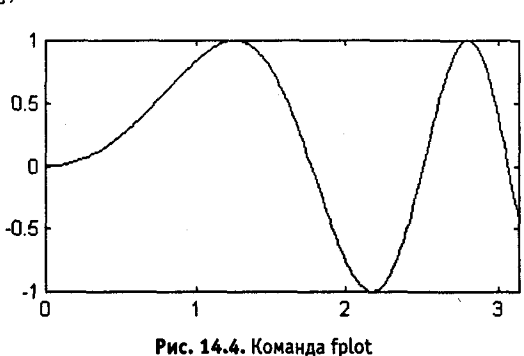


Рис. 14.4. Команда fplot

Для изображения нескольких рисунков в одном окне применяется команда `subplot` (разделение окна на подокна). Эта команда позволяет организовать массив графиков, состоящий из `N` рисунков по вертикали и `M` рисунков по горизонтали. В результате выполнения команды

```
subplot(N,M,K)
```

активным становится окно с номером K, причем нумерация ведется слева направо начиная с верхнего ряда. Крайний слева рисунок из верхнего ряда считается первым, а крайний справа из нижнего ряда рисунок имеет номер N*M. Пример использования команды subplot дан ниже в подразделе «Масштабирование» раздела «Оформление рисунка».

Оформление рисунка

В последних реализациях MATLAB (версия 5.3 и 6.0) расширены средства оформления графиков непосредственно в графическом окне при помощи клавиш мыши, пунктов меню окна и контекстных меню, возникающих при наведении мыши на элементы рисунка. Это позволяет дорисовывать линии, проводить стрелки, изменять толщину, цвет и тип линии, маркер и его размер, дописывать текст. Изображения трехмерных и двумерных рисунков можно повернуть интерактивно, распечатать и сохранить в одном из графических форматов, см. подраздел «Интерактивная работа с графикой». Все эти действия и ряд других могут быть реализованы также при помощи команд оформления, запускаемых в командном окне.

Надписи и маркировка

Для оформления графиков используются команды, позволяющие снабдить рисунок заголовком, вывести имена переменных по осям, сделать в нужном месте надпись и т. д.

Таблица 14.4. Команды оформления рисунка

Имя	Назначение
title('TEXT')	Вывод заголовка TEXT
xlabel('TEXT')	Маркировка оси x
ylabel('TEXT')	Маркировка оси y
text(x,y,'TEXT')	Вывод текста в заданное координатами (x,y) место графика
gtext('TEXT')	Вывод текста в интерактивно определяемое место на графике (указать место мышкой и щелкнуть)
legend	Идентификация кривых (легенда)
grid	Нанесение координатной сетки
box	Рисование рамки
hold	Резервирование рисунка для последующего вывода
ishold	Запрос о резервировании рисунка (1 — да, 0 — нет)

По умолчанию на двумерных рисунках координатная сетка не наносится. Для задания сетки нужно выполнить команду grid on, а убрать сетку можно по команде grid off. Команда grid без параметров действует как переключатель: устанавливая режим сетки и убирая его. Аналогичным синтаксисом обладает команда рисования рамки box.

Если в графическом окне имеется рисунок и выполняется новая графическая команда рисования, то старое изображение замещается новым, потому что действует режим обновления — `hold off`, установленный по умолчанию. Чтобы запретить стирание предыдущего изображения при выводе нового, нужно выполнить команду `hold on`. Команда `hold` без параметров действует как переключатель. Команда `ishold` позволяет узнать, включен ли для текущего окна (Figure) режим сохранения (резервирования) изображения.

Выводимый текст может содержать конструкции TeX (см. главу 19 «Краткое введение в пакет LaTeX»). Это позволяет использовать для маркировки осей и рисунка греческий алфавит, индексы и степени, а также математические символы. При подготовке иллюстраций желательно, чтобы пакет использовал текущие значения параметров для описания рисунка. Для перевода численных данных в строковые переменные применяется команда `num2str`, см. главу 12 «Элементы языка MATLAB».

Следующий пример (см. рис. 14.5) демонстрирует различные способы нанесения надписей, а также изменение размера символов:

```
» t=0:.2:14; gam=exp(-t/9).*cos(t); plot(t,gam)
xlabel("Время",'Fontsize',12),
t1=6.4; gam1=gam(find(t==t1));
s1=num2str(t1); s2=num2str(gam1);
s=strcat("\leftarrow \Gamma_1(",s1,")=',s2);
text(t1,gam1,s,'Fontsize',10);
text(-1.7,0, "\Gamma_1",'Fonts',14)
```

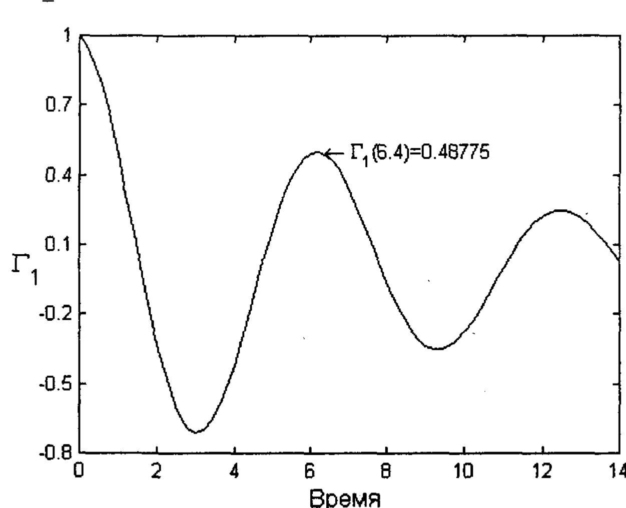


Рис. 14.5. Маркировка осей и нанесение поясняющих надписей

В этом примере для нанесения текста использована символьная переменная `s`, полученная объединением четырех других при помощи команды `strcat`. Строки `s1` и `s2` получены преобразованием чисел `t1` и `gam1`. Для маркировки оси абсцисс использована стандартная команда `xlabel`, а для маркировки оси ординат применена команда `text` со строкой, содержащей заглавную греческую букву и индекс, оформленные согласно стандарту TeX.

Команда нанесения надписей допускает следующую интересную возможность. Если приготовить строковую переменную той же размерности, что и массивы, указывающие местоположение надписи, то для каждой точки будет выведен символ из строковой переменной:

```
» s='MATLAB & Maple + LaTeX'; y=double(s);
x=linspace(1,length(y),length(y)); plot(x,y,':'),
text(x,y,s','Fontsize',14)
```

В результате получим:

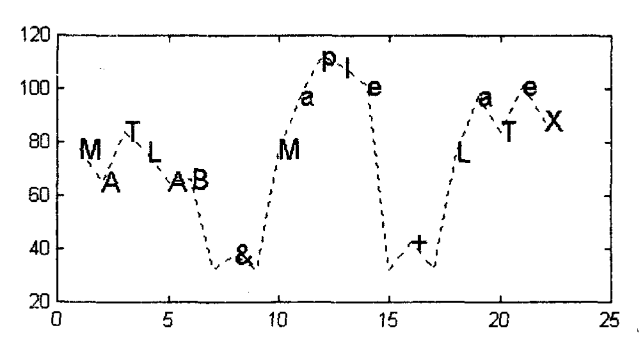


Рис. 14.6. Команда text позволяет делать букет надписей

Приведенный пример экзотичен, но часто бывает нужно выделить одну или несколько частей заданной кривой. Конечно, можно разбить кривую на участки и нарисовать их отдельно, а можно в соответствии с некоторым условием сформировать символьный вектор и затем вывести график как надпись, где разные участки будут помечены своими символами.

Масштабирование

При построении графика выбор масштаба и построение осей совершаются автоматически, а для изменения масштабов применяется команда axis. Чтобы определить интервалы изменения координат самостоятельно, нужно выполнить команду

```
axis([xmin,xmax,ymin,ymax])
```

Здесь числа xmin и xmax задают интервал изменения горизонтальной координаты (минимальное и максимальное значения), а ymin и ymax — соответственно интервал изменения вертикальной координаты. Если нужно сохранить автоматическое масштабирование по какой-нибудь оси, то в качестве значения следует поставить идентификатор Inf (или -Inf), см. команды к рис. 14.12. Характерные применения команды axis сведены в табл. 14.5.

Таблица 14.5. Варианты обращения к команде axis

Команда	Действие
axis(axis)	Использование текущего масштаба для последующих графиков (фиксация текущих назначений)
axis(auto)	Восстановление режима автомасштабирования

продолжение ⇗

Таблица 14.5 (*продолжение*)

Команда	**Действие**
v=axis	Получение вектора с текущими значениями масштаба
axis('ij')	Размещение начала отсчета в левом верхнем углу (матричная система координат)
axis('xy')	Размещение начала отсчета в левом нижнем углу (декартова система координат)
axis off	Отключение обозначения осей и насечек
axis on	Восстановление осей и насечек

Для коррекции размеров рисунка используются стили масштабирования, позволяющие установить одинаковый масштаб по обеим переменным и согласовать область вывода. Для указания нужного стиля STYLE нужно выполнить команду

```
axis STYLE
```

Таблица 14.6. Стили масштабирования для команды axis

Имя	**Описание**
square	Область вывода — квадрат
normal	Масштабирование по умолчанию. MATLAB выбирает диапазон и разметку по собственному алгоритму
equal	Одинаковый масштаб по осям
tight	Изображаемые данные занимают всю область вывода
auto	Включение автомасштабирования
manual	Масштабирование вручную

Применение различных стилей масштабирования для изображения эллипса демонстрирует следующий пример:

```
» t=0:pi/30:2*pi;x=2.8*sin(t);y=1.9*cos(t);
subplot(2,3,1),plot(x,y), text(-1,0,'normal')
subplot(2,3,2),
                              plot(x,y), axis([-3 3 -2 2]), text(-1,0,'manual')
subplot(2,3,3),
                              s='square'; plot(x,y), axis(s), text(-1,0,s)
subplot(2,3,4), plot(x,y),axis equal, text(-1,0,'equal')
subplot(2,3,5), plot(x,y),axis tight, text(-1,0,'tight ")
subplot(2,3,6),
plot(x,y), axis equal tight, text(-1,0,'equal tight')
```

На рис. 14.7 в разных окнах представлены изображения эллипса, а в центре каждого рисунка дана надпись, соответствующая использованному стилю масштабирования. Показано также, что для задания стиля можно использовать строковую переменную. Так, слово square присвоено переменной s, и для крайнего правого подокна верхнего ряда эта переменная определяет стиль масштабирования при помощи команды axis и выдаваемый текст (команда text).

Для того чтобы выяснить текущие назначения команды axis, можно ввести

```
[s1,s2,s3]=axis("state")
```

По умолчанию действует режим ["auto",'on','xy'].

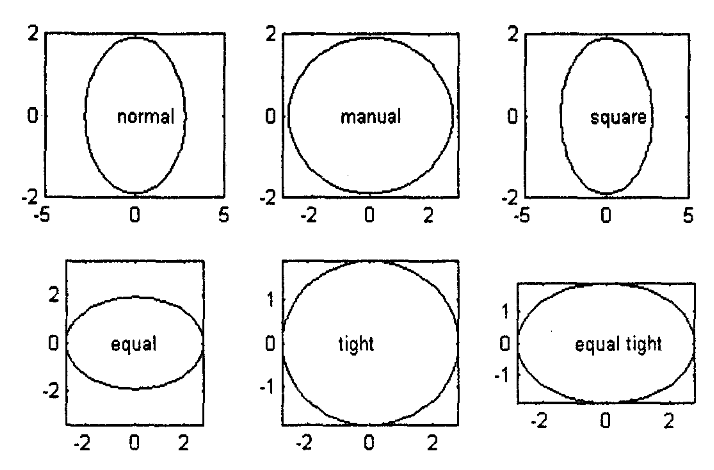


Рис. 14.7. Разные стили масштабирования

Элементы дескрипторной графики

Функции системы дескрипторной графики Handle Graphics позволяют эффективно работать с графическими объектами (линиями, поверхностями и другими объектами). Основными здесь являются исходные одиннадцать примитивов, перечисленные в табл. 14.7.

Таблица 14.7. Перечень базисных примитивов

Объект	Назначение
Root	Корневой объект, соответствующий экрану и создаваемый MATLAB автоматически в начале сеанса
Figure	Графическое окно на экране
Uicontrol	Пользовательский интерфейс (кнопки, прокрутки и т. д.)
Axes	Физическая область в окне
Uimenu	Пользовательское меню
Image	Двумерный образ, задаваемый массивом
Line	Базисный примитив линии
Patch	Базисный примитив закрашенного многоугольника
Surface	Базисный примитив поверхности
Text	Строки символов
Light	Источники света, действующие на объекты в пределах области, определенной Axes

Например, объект Line нуждается в размерах отображаемой области, которые содержатся в Axes. В свою очередь, для Axes требуются данные об объекте Figure.

Каждый графический объект при его создании получает некоторое значение (дескриптор), которое присваивается идентификатору и впоследствии может быть использовано для изменения свойств объекта. Если график сформирован из нескольких объектов, то каждый объект характеризуется своим дескриптором.

Значение дескриптора для корневого объекта равно нулю, а для окна (Figure) совпадает с его номером. Для остальных объектов дескрипторы задаются вещественными числами, содержащими информацию, используемую MATLAB. Для использования дескриптора графического объекта необходимо присвоить результат выполнения графической команды некоторой переменной. Эта переменная является дескриптором и предоставляет доступ к свойствам графического объекта. Информацию о дескрипторе графического объекта позволяет получить функция get, а функция set служит для переопределения свойств объекта.

Например, пусть матрица A состоит из трех столбцов:

```
» A=[1 2 3; 1 0 -1]
A =
     1     2     3
     1     0    -1
```

Тогда по команде

```
» h=plot(A)
```

будет организовано графическое окно (figure), проведены три ломаные разного цвета и выданы значения переменной h:

```
h =
    1.0010
   74.0007
   75.0002
```

Для получения полной информации о первой кривой следует ввести get(h(1)), а для вывода цвета достаточно задать:

```
» get(h(1),'Color')
ans =
     0     0     1
```

Списки параметров различных объектов достаточно объемны и здесь не приводятся.

Чтобы переопределить свойство объекта при помощи функция set, достаточно указать несколько первых букв, специфицирующих свойство объекта. Например, вместо полных имен Color и LineWidth допустимо использовать следующие сокращения:

```
» set(h(1),'Col',[0 .8 .8],'LineW',3)
```

Чтобы получить перечень возможных значений какого-нибудь свойства Prop, надо выполнить команду set(h,'Prop'). Например:

```
» set(h(2),'LineStyle')
[ {-} | -- | : | -. | none ]
```

В табл. 14.8 приведены функции, предоставляющие доступ к часто используемым дескрипторам. Отметим, что значения дескрипторов могут меняться в зависимости от платформы и сеанса.

Эти функции можно использовать в качестве аргументов других функций. Например, удалить графическое окно можно по команде

```
» delete(gcf)
```

Таблица 14.8. Команды доступа к дескрипторам графических объектов

Имя	Назначение
gcf	Получение дескриптора текущего окна (figure)
gca	Получение дескриптора осей (axes) для текущего окна (figure)
gco	Получение дескриптора текущего объекта для текущего окна (figure)

Для очистки текущего окна достаточно выполнить команду clf, а для осей — cla.

Приведем пример построения графика с использованием специальных отметок по оси абсцисс:

```
» days=["Пон";'Вто';'Сре';'Чет']; temp=[1 3 7 5]';
f=figure; a=axes("Ylim",[0 8],'Xtick',1:4,'Xticklab',days);
h=line(1:4,temp,'LineW',2);
```

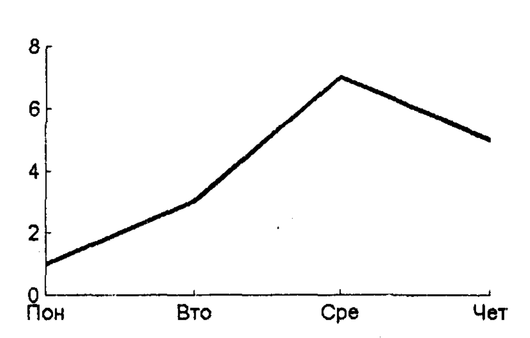


Рис. 14.8. Построение кривой с использованием дескрипторной графики

Три дескриптора объектов Figure, Axes, Line, — соответственно переменные f, a, h — определены для последующего использования.

Следующий пример демонстрирует возможность изменения оформления осей координат, что обеспечивает появление на оси ординат трех участков с одинаковой маркировкой. Вначале нарисуем три функции Бесселя со сдвигом по оси ординат на две единицы:

```
» x=0:.2:12; te='Функция Бесселя J_'; hold on
for k=0:2, y=besselj(k,x); plot(x,2*k+y),
s=strcat(te,num2str(k)); text(5,2*k+.4,s), end
```

Выведем текущие назначения осей. Ниже список выдаваемых параметров существенно сокращен, а вместо пропущенных строк стоит многоточие:

```
» get(gca)

                    AmbientLightColor = [1 1 1]
                    Box = off
. . .
                    FontName = Helvetica
                    FontSize = [10]
. . .
                    YColor = [0 0 0]
```

```
YDir = normal
YGrid = off
YLabel = [81.002]
YAxisLocation = left
YLim = [-0.5 4.5]
YLimMode = auto
YScale = linear
YTick = [ (1 by 11) double array]
YTickLabel = [ (11 by 4) char array]
```

. . .

После этого можно запустить команду переназначения меток насечек, подставив свою последовательность чисел для параметра YtickLabel вместо автоматически сгенерированного возрастающего числового ряда

```
» set(gca,'YtickLabel',[-.5 0 .5 1])
```

В результате на рис. 14.9 каждой кривой будет отвечать отрезок оси ординат с правильными значениями для соответствующей функции Бесселя.

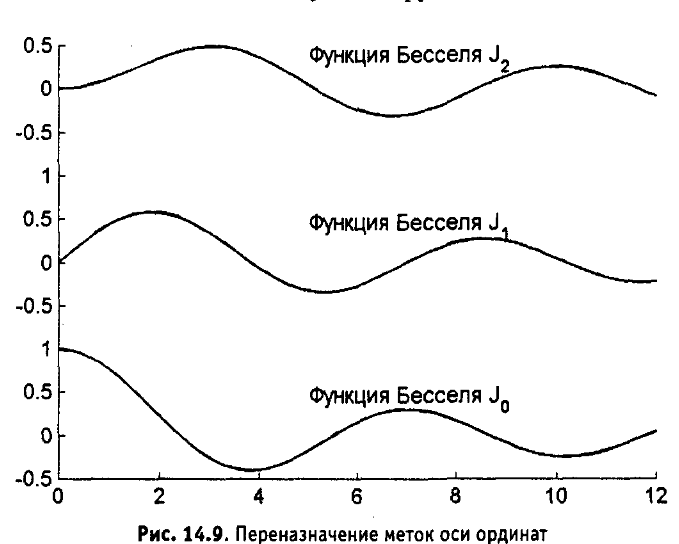


Рис. 14.9. Переназначение меток оси ординат

Трехмерная графика

Команда plot3 для построения кривых в трехмерном пространстве похожа на рассмотренную команду двумерной графики plot. Кривая строится в перспективной проекции по трем векторам одинаковой размерности x, y и z:

```
plot3(x,y,z)
```

Можно нарисовать кривую при помощи массива xyz с тремя столбцами, где каждый столбец хранит данные о своей координате:

```
plot3(xyz)
```

Применение команды plot3 для построения траектории решения системы Лоренца дано в подразделе «Интегрирование дифференциальных уравнений» главы 15 «Численный анализ в MATLAB».

Если требуется нарисовать несколько кривых, то возможны следующие решения:

- приготовим три двумерных массива X, Y, Z, каждый из которых содержит данные об одной координате. Столбцы этих массивов отвечают разным кривым, то есть первая кривая задается первыми столбцами массивов X, Y, Z, вторая — вторыми столбцами и т. д. В этом случае обращение к команде plot3 имеет вид:

```
plot3(X,Y,Z)
```

- второй способ заключается в последовательном перечислении векторов, содержащих координаты кривых:

```
plot3(x1,y1,z1,s1,x2,y2,z2,s2, ...)
```

Здесь сочетаниями s1, s2, ... обозначены маркировки кривых (тип линии, маркер, цвет), например строка 'y-.' задает построение пунктирной кривой желтого цвета и точкой в качестве маркера.

Команды оформления axis, title, xlabel и другие действуют аналогично командам, рассмотренным для двумерной графики.

Построение поверхностей

Чтобы построить поверхность, нужно иметь массив значений функции нескольких переменных, вычисленный на некоторой сетке. Для формирования двумерной прямоугольной и трехмерной параллелепипедальной сеток используется команда meshgrid, а для функций с большим числом переменных — команда ndgrid.

Пусть x, y — одномерные массивы точек, задающие абсциссы и ординаты соответственно. В результате выполнения команды

```
[X,Y]=meshgrid(x,y)
```

будут сформированы два двумерных массива X и Y, причем все строки массива X будут копиями вектора x, а столбцы Y — копиями вектора y. Теперь с помощью этих массивов можно вычислить массив значений функции двух переменных и затем нарисовать поверхность. В MATLAB для изображения поверхности имеются команды, перечисленные в табл. 14.9.

Таблица 14.9. Команды построения поверхности

Команда	Назначение
mesh	Построение сетчатой поверхности
meshc	Построение сетчатой поверхности с линиями уровня
meshz	Построение сетчатой поверхности и отсчетной плоскости
surf	Построение расцвеченной поверхности
surfc	Построение расцвеченной поверхности с линиями уровня
surfl	Построение расцвеченной поверхности с подсветкой
waterfall	Трехмерная поверхность без прорисовки ребер

Команда mesh строит в графическом окне расцвеченную сетчатую поверхность, используя различную окраску вершин и ребер. Чтобы ребра были окрашены в цвета прилегающих вершин (в процессе обхода вершин цвет ребра меняется), нужно выполнить команду

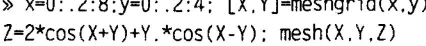
```
shading flat
```

а для плавного изменения цвета между вершинами (линейная интерполяция) надо запустить команду

```
shading interp
```

Приведем пример построения сетчатой поверхности, см. рис. 14.10.

```
» x=0:.2:8;y=0:.2:4; [X,Y]=meshgrid(x,y);
```



```
Z=2*cos(X+Y)+Y.*cos(X-Y); mesh(X,Y,Z)
```

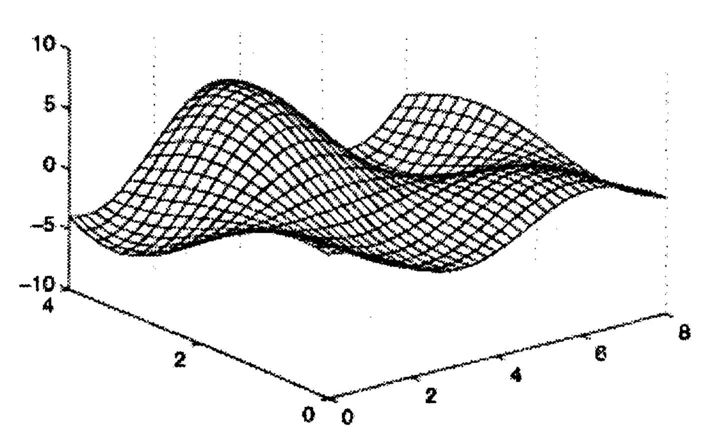


Рис. 14.10. Построение поверхности по команде mesh

Та же сетчатая поверхность будет построена при помощи команды mesh(x,y,Z).

Имеется несколько разновидностей команды mesh. На рис. 14.11 слева нарисована поверхность -Z с линиями уровня (команда meshc), а справа та же поверхность — с отсчетной плоскостью (команда meshz):

```
» subplot(1,2,1), meshc(x,y,-Z), shading interp
subplot(1,2,2), meshz(x,y,-Z), shading flat
```

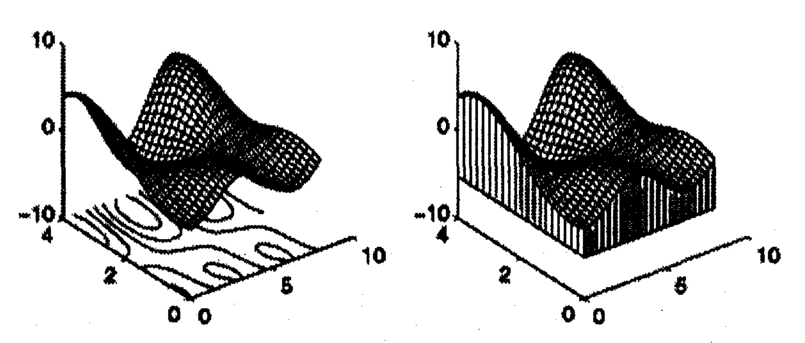


Рис. 14.11. Построение поверхностей командами meshc и meshz

Ряд команд для построения поверхностей использует затенение. На рис. 14.12 приведены три варианта: затененная сеточная поверхность (слева), то же с линиями уровня (в центре) и поверхность с подсветкой (справа), построенные с помощью команд

```
» subplot(1.3.1). surf(x.y.Z).
subplot(1.3.2). surfc(x.y.Z).
subplot(1.3.3). surfl(x.y.Z)
```

После построения рисунков для каждого подокна выполнено масштабирование при помощи команды

```
» axis([-Inf Inf -Inf Inf -Inf Inf])
```

В результате были определены действительные интервалы изменения величин по всем координатам и рисунки стали выразительнее (сравните с рис. 14.10). Изменения не коснулись только нижней границы по координате z для среднего рисунка, поскольку плоскость с линиями уровня изображается при том минимальном значении z, которое MATLAB выбрал сам.

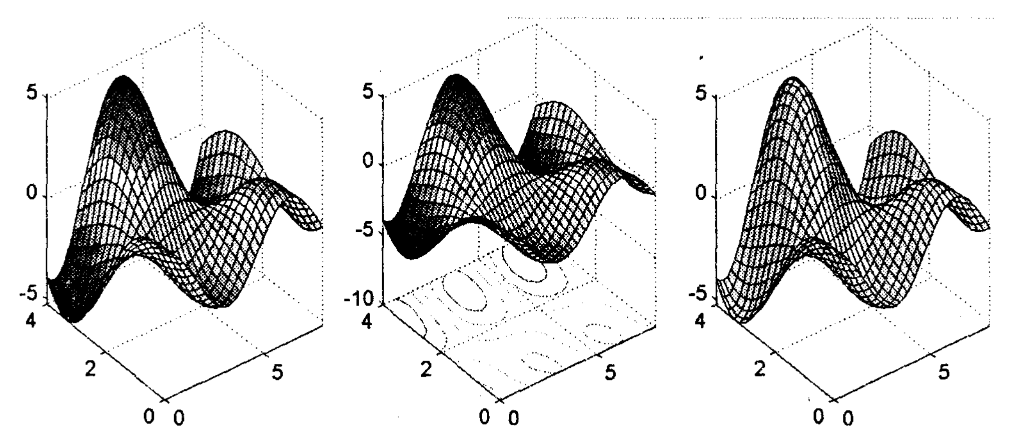


Рис. 14.12. Использование команд surf, surfc, surfl для построения поверхностей

Когда поверхность построена, то для получения нужного ракурса можно воспользоваться командой view, устанавливающей угол зрения наблюдателя по отношению к осям

```
view(AZ.EL) или view([AZ.EL])
```

Здесь AZ — угол вращения в горизонтальной плоскости (азимут), а EL — угол возвышения. Угол зрения можно задать также посредством декартовых координат:

```
view([x.y.z])
```

причем важны отношения величин, а не их абсолютные значения. Команда view без параметров выведет матрицу поворота размера 4×4.

Имеется возможность выбрать ракурс интерактивно средствами графического окна. Нужно установить режим вращения изображения при помощи пункта **Rotate 3D** меню **Tools** или соответствующего значка, а затем, перемещая мышь при нажатой

клавише, задать расположение осей. При этом в левом нижнем углу окна будут отображаться значения углов AZ и EL.

Приведем пример использования команды view и установки нужного ракурса для построенной командой waterfall поверхности без прорисовки поперечных ребер, см. рис. 14.13:

```
» x=1:.1:4;y=0:.02:1; [X,Y]=meshgrid(x,y);
Z=.2*X+(1-Y).*sin(pi*X.*Y);
waterfall(X',Y',Z'), axis equal, view(-140,30), grid
```

В этом примере установлен единый масштаб по координатам x и y и использована графическая команда grid, чтобы убрать нанесение сетки.

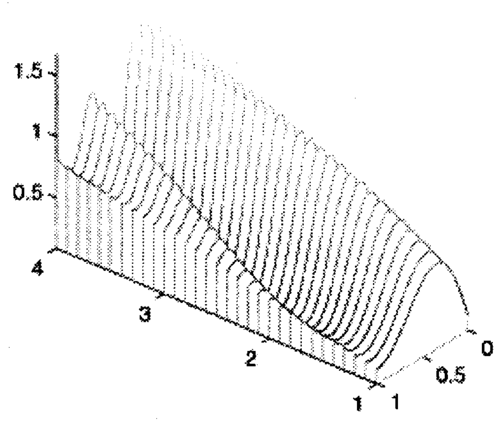


Рис. 14.13. Построение поверхности без поперечных ребер (команда waterfall)

Функция ndgrid формирует массивы для вычисления и интерполяции функций нескольких переменных. Подобно команде meshgrid, на вход подается несколько одномерных векторов x1, x2, ..., а на выходе имеем многомерные массивы X1, X2, ..., причем k-я размерность массива Xk есть копия вектора xk. Рассмотрим использование команд ndgrid и slice для построения сечений функции трех переменных, см. рис. 14.14:

```
» x1=-2:.2:2; x2=-2:.2:2; x3=0:.05:1;
[X1,X2,X3]=ndgrid(x1,x2,x3); V=(X3-.3).*sin(X2 + X1);
slice(x1,x2,x3,V,[-.8 .6 2],2,[0 .5])
xlabel("x1"), ylabel("x2"), zlabel("x3")
```

Для оформления трехмерной графики используются команды, аналогичные рассмотренным в разделе «Оформление рисунка» этой главы, и добавившаяся команда zlabel для маркировки третьей оси. В табл. 14.10 дана сводка этих команд.

Таблица 14.10. Команды оформления рисунка

Имя	Назначение
axis	Масштабирование
grid	Нанесение сетки

Имя	Назначение
legend	Пояснения к графику
text('text')	Вывод текста в заданное место графика
title('text')	Вывод заголовка
xlabel('text')	Маркировка оси x
ylabel('text')	Маркировка оси y
zlabel('text')	Маркировка оси z
gtext('text')	Вывод текста в интерактивно определяемое место на графике (указать место мышкой и щелкнуть)

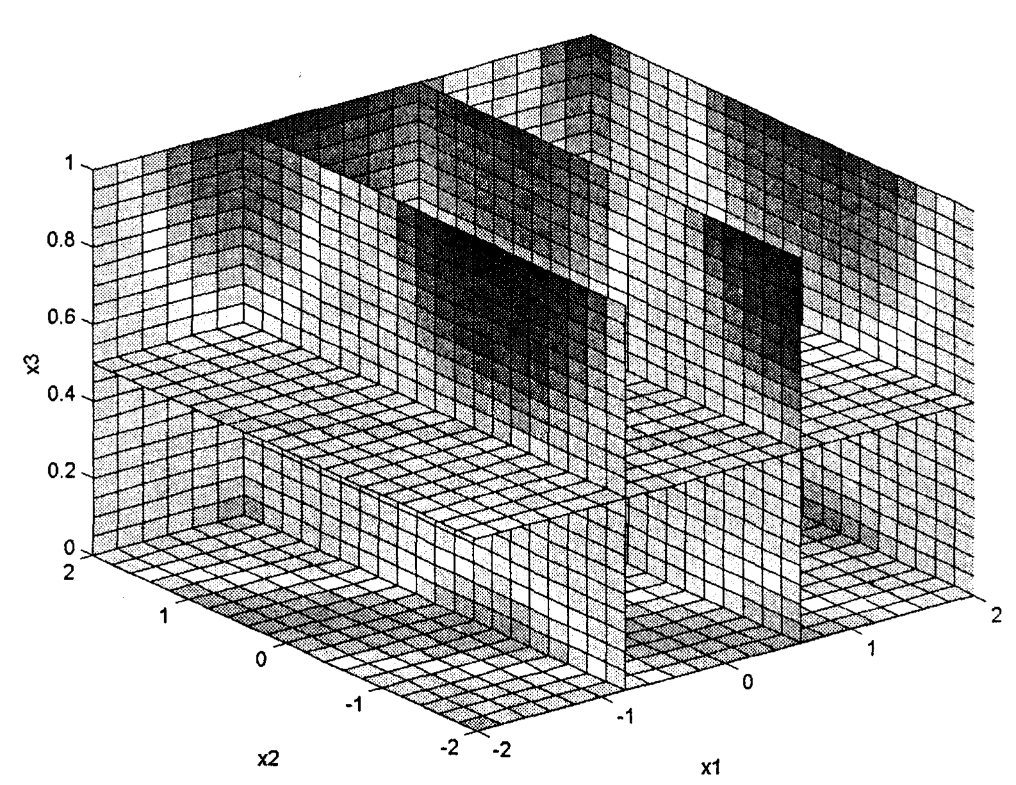


Рис. 14.14. Построение сечений для функции нескольких переменных (команда slice)

Палитра и подсветка

Использование цветовой гаммы для оформления рисунка повышает его информативность, позволяет создавать качественные иллюстрации для публикаций и размещения в Интернете. Рассмотрим коротко команды работы с цветом в MATLAB.

Каждый рисунок (окно Figure) имеет свою палитру, заданную матрицей с тремя столбцами и числом строк, отвечающим количеству заданных цветов. Три числа

на одной строке, которые могут принимать значения от 0 до 1, определяют соответственно интенсивности красной, зеленой и голубой компонент (RGB).

Для задания и изменения палитры используется команда colormap. По умолчанию стандартной является палитра jet, содержащая 64 цвета. Чтобы назначить новую палитру при помощи массива CC, нужно выполнить команду colormap(CC), а вернуть стандартную палитру можно при помощи команды colormap('default'). Обращение к команде colormap без параметров выводит матрицу с текущей палитрой. Например, желтому цвету отвечает 41 строка:

```
» cm=colormap: cm(41,:)
ans =
    1    1    0
```

Используя стандартные операции с массивами, можно приготовить свою палитру. В то же время имеется набор приготовленных палитр, в частности: hsv (радуга), hot (комбинации черного, красного, желтого и белого), cool (фиолетово-голубая палитра), summer (желто-зеленая палитра), bone (серо-синяя палитра), gray (оттенки серого). При обращении к любой из этих функций указывается параметр, задающий число цветов, а если параметр не указан, то формируется палитра с тем же числом цветов, что и текущая. Например, по команде colormap(gray(4)) сформируется палитра из четырех оттенков серого цвета.

Примеры формирования цветов посредством указания долей красного, зеленого и синего представлены в табл. 14.11.

Таблица 14.11. Формирование цветов

Цвет	Red	Green	Blue	Color
черный	0	0	0	black
белый	1	1	1	white
красный	1	0	0	red
зеленый	0	1	0	green
синий	0	0	1	blue
желтый	1	1	0	yellow
фиолетовый	1	0	1	magenta
голубой	0	1	1	cyan
серый	.5	.5	.5	gray
ярко-красный	.5	0	0	dark red
медный	1	.62	.4	copper
аквамарин	.49	1	.83	aquamarrine

Палитры необходимы при построении графических объектов: поверхностей с помощью команд mesh, surf и др., контурных рисунков (смотри далее о семействе команд contour), и т. д. Диапазон изменения цвета для текущей палитры можно сузить при помощи команды

```
caxis([cmin cmax])
```

Здесь cmin и cmax определяют диапазон палитры (соответственно начальный и конечный цвета), который будет использоваться для окрашивания. Команда caxis без параметров выводит текущие значения cmin и cmax, команда caxis("auto") позволяет вернуться к автоматическому назначению цветов.

Использование подсветки придает больший реализм изображаемым трехмерным объектам. Для этого в MATLAB имеется графический объект Light, создать который можно при помощи команды light. Подсветка действует на объекты, построенные при помощи команд surface, patch, surf, mesh, fill, fill3. Управление эффектом подсветки производится путем указания свойств объектов Axes, Light, Patch и Surface.

Перечислим основные свойства объекта Light:

- Color — цвет источника света;
- Style — параметр, указывающий бесконечную (по умолчанию) или конечную удаленность (значение local) источника света от объекта;
- Position — три координаты, задающие положение источника света в случае конечной его удаленности или направление на бесконечно удаленный источник света. Направление во втором случае определяется прямой, проходящей через эту точку и начало координат.

Для ознакомления с другими свойствами, влияющими на эффект подсветки, нужно обратиться к справочной системе MATLAB. Отметим, что поля структуры имеют длинные имена, однако для спецификации поля достаточно набрать несколько начальных символов имени, отличающих это имя от других.

Приведем пример различного освещения сферы для иллюстрации применения подсветки. Покажем также, как управлять свойствами графического объекта surface. Определим красный цвет для всей сферы и будем менять в цикле два параметра, задающих свойства подсветки, — SpecularStrength и DiffuseStrength (допустимое сокращение Diffuse), а также положение источника света (команда light). Определим для всех рисунков одинаковый масштаб, убрав маркировку осей и нанесение сетки, производимые по умолчанию:

```
» n=3; km=1;
for k=1:n, for m=1:n, subplot(n,n,(k-1)*n+m),
sphere(60); h=findobj("Type",'surface');
set(h(km),'CdataM','direct','EdgeColor',[1 0 0]);
cc=get(h(km),'Cdata');
set(h(km),'Cdata',57*ones(size(cc)));
set(h(km),'EdgeLighting','flat','FaceLight','phong')
set(h(km),'SpecularStrength',1.0*(k-1))
set(h(km),'Diffuse',0.4*(m-1)+0.2)
grid off, axis equal, axis off
z=.75; axis([-z z -z z -z z])
light("Position",[-4 (1-m)*1.5 (m-1)*1.5])
end, end
```

Результаты приведены на рис. 14.15.

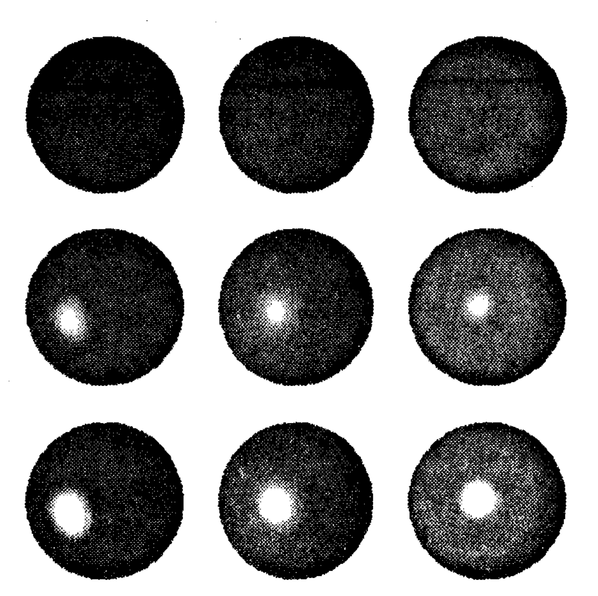


Рис. 14.15. Действие различных режимов подсветки

Специализированная графика

Команды построения двумерных и трехмерных кривых, поверхностей и команды оформления рисунков не исчерпывают всех графических возможностей MATLAB. Приведем в табл. 14.12 перечень некоторых команд специализированной графики, полный список которых можно получить по справке

```
help specgraph
```

Таблица 14.12. Команды специализированной графики

Имя	Назначение
bar и bar3	Построение столбцовой диаграммы
pie и pie3	Построение круговой диаграммы
fill и fill3	Рисование цветного многоугольника
hist	Рисование гистограммы
polar	График в полярных координатах
quiver	Вывод векторного поля
rose	Гистограмма в полярных координатах
stairs	Ступенчатый график
stem и stem3	Дискретный график («стебель»)

Графики синуса, построенные при помощи команд bar, stem и stairs, будут выглядеть следующим образом:

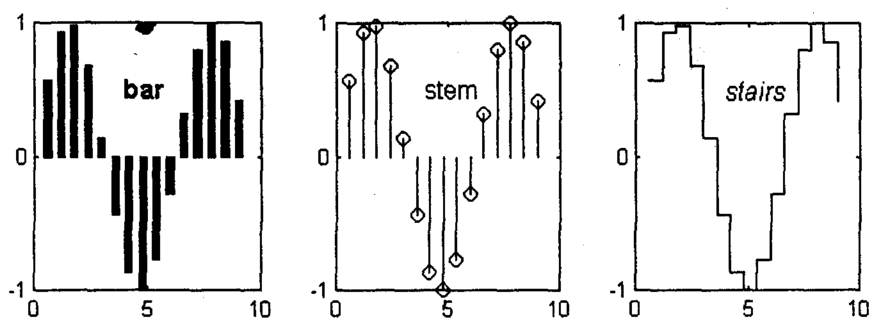


Рис. 14.16. Команды специализированной графики bar, stem, stairs

Для построения были выполнены операторы, приведенные ниже:

```
» x=[0.6:.6:9];s=sin(x);
subplot(131), bar(x,s,.5), text(4,.5,'bar')
subplot(132), stem(x,s), text(4,.5,'stem')
subplot(133), stairs(x,s), text(4,.5,'stairs')
```

Стиль надписей был изменен с помощью пункта Font Style контекстного меню графического окна, см. раздел «Интерактивная работа с графикой» этой главы.

Для иллюстрации трех других команд специализированной графики создадим структуру и конвертируем ее в числовой массив:

```
» t={"Зима",'Весна','Лето','Осень'}; p=double(char(t));
subplot(131), pie(p(:,4),t), subplot(133), bar3(p),
subplot(132), pie3(p(:,4),[1 1 1 1])
```

Результат выполнения этих команд приведен на рис. 14.17, причем в данном случае расположение надписей было изменено в графическом окне при помощи мыши.

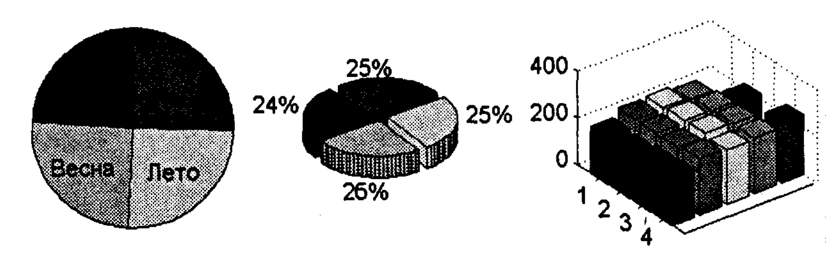


Рис. 14.17. Результат выполнения команд pie, pie3 и bar3

Отметим, что пример построения векторного поля при помощи команды quiver дан в главе 15 «Численный анализ в MATLAB». Для иллюстрации рисования с использованием команды stem3 и в полярных координатах приведем рис. 14.18, полученный в результате выполнения следующих команд:

```
» t=0:.2:12; s=0.1+i; y=exp(-s*t); ry=real(y); iy=imag(y);
subplot(121), polar(ry,iy,'k'),
subplot(122), stem3(ry,iy,t,'k'), hold, plot3(ry,iy,t,'k')
```

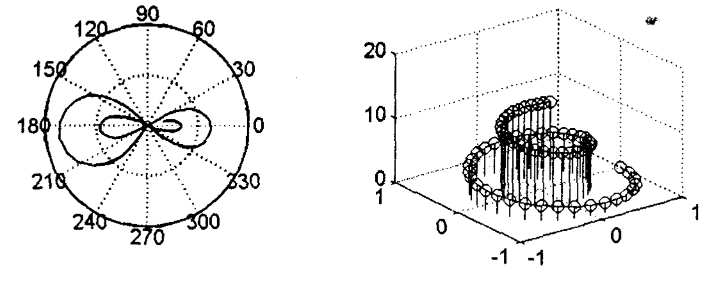


Рис. 14.18. Применение команд polar (слева), stem3 и plot3 (справа)

Как видно, в MATLAB на одном рисунке можно совмещать разные виды графиков. Приведем еще один пример и продемонстрируем, как изменить маркировку осей. Вычислим ранг магических матриц размера от 1 до 33 и посчитаем логарифм от величины размерности матрицы и поместим данные в массивы r и s соответственно. Напомним, что магической называется матрица с одинаковой суммой элементов по строкам, столбцам и диагоналям. Построим столбцовую диаграмму для массива r и наложим поверх нее кривую s:

```
» k=1:33; r=zeros(size(k)); s=zeros(size(k));
for m=k, M=magic(m); r(m)=rank(M);s(m)=log(m); end,
bar(k,r,.6);
xlabel("m"), ylabel("Rank"), title("Magic matrix")
```

Сохраним сведения о свойствах осей (объект Axes) построенной диаграммы в переменной h1 и создадим новые оси. Затем нарисуем график:

```
» h1=gca; h2=axes("Position",get(h1, "Position"));
plot(k,s,'LineWidth',2),
```

Чтобы новые оси не перекрывали старые, разместим их справа, уберем насечки с оси абсцисс и выведем их сверху. Наконец, надпишем новую ось ординат:

```
» set(h2,'YAxisLocation','right','Color','none')
set(h2,'XTickLabel',[],'Xlim',get(h1,'Xlim'),'Layer','top')
ylabel("Log(m)"),
```

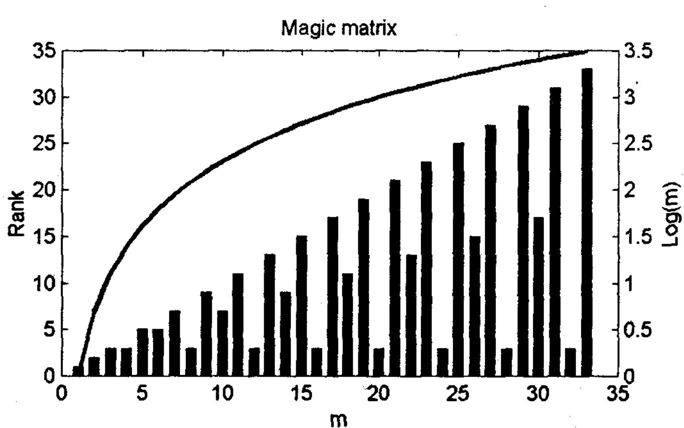


Рис. 14.19. Совмещение разных типов графики на одном рисунке

Теперь перечислим ряд команд построения объемных фигур и для их описания используем следующие обозначения: R — вектор радиуса, N — число разбиений окружной координаты, массивы X, Y, Z определяют векторы координат, цвет задается массивом C, TRI — массив треугольников.

Таблица 14.13. Построение объемных фигур

Команда	Назначение
[X,Y,Z]=cylinder(R,N)	Цилиндр
[X,Y,Z]=sphere(N)	Сфера
trimesh(TRI,X,Y,Z,C)	Поверхность из треугольников
trisurf(TRI,X,Y,Z,C)	Поверхность из треугольников без прорисовки сетки

Пример построения сферы дан на рис. 14.15.

Имеются команды для рисования трехмерных объектов: и для перехода от декартовой системы координат к сферической (cart2sph), и наоборот (sph2cart).

Линии уровня

Для изображения линий уровня функции двух переменных, представленной двумерным массивом значений, применяется команда contour. Пусть массив Z определяет значения функции, вычисленные в определенных массивами X и Y точках. Рассмотрим несколько вариантов возможного использования команды contour:

```
contour(X,Y,Z)
```

Число линий уровня и их значения при таком вызове выбираются автоматически. Если X и Y отсутствуют, то при обращении contour(Z) сетка считается квадратной. При помощи дополнительного целого параметра N можно задать число линий уровня:

```
contour(X,Y,Z,N)
```

Чтобы построить линии уровня при требуемых значениях рассматриваемой функции, следует определить массив соответствующих величин V:

```
contour(X,Y,Z,V)
```

При этом число линий уровня равно length(V). Этот способ применения команды contour позволяет, например, нарисовать одну линию уровня при заданном скалярном значении v: contour(X,Y,Z,[v v]).

Если результат выполнения команды contour присвоить переменным [C,H]=contour(X,Y,Z), то полученную матрицу C и столбец H можно использовать в качестве входных параметров для команды clabel(C,H), которая выведет цифровую информацию о значениях линий уровня изображаемой функции. Для окраски линий и областей между ними применяются стандартные назначения, изменить которые можно при помощи указания параметра S для определения типа, цвета и толщины линий подобно тому, как это делается для команды plot: contour(X,Y,Z,'S').

Имеются еще две команды построения линий уровня, обращение к которым аналогично вызову функции contour: команда contourf позволяет закрашивать облас-

ти между линиями равного уровня, а команда contour3 изображает линии уровня в трехмерном пространстве.

Сопутствующими для команд построения линий уровня является также команда вывода цветовой шкалы colorbar.

Приведем на рис. 14.20 различные варианты изображения линий уровня, полученные в результате выполнения следующих команд:

```
» x=0:.1:4; y=0:.1:2; [X,Y]=meshgrid(x,y);
Z=cos(2*X+Y)+Y.*cos(X-Y); colormap(gray), cc=[-2 0 4];
subplot(1,3,1), contourf(x,y,Z, [-6:1:6]); colorbar
subplot(132),[c,h]=contour(X,Y,Z,cc,'-k'); clabel(c,h),
subplot(1,3,3), C3=contour3(x,y,Z,32); grid off
```

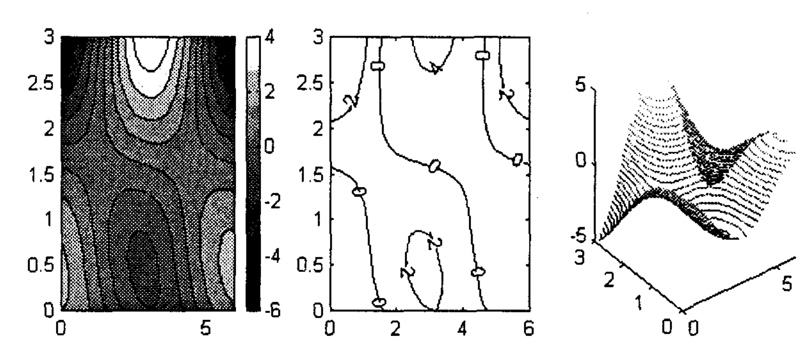


Рис. 14.20. Построение линий уровня

Анимация

Многие явления и эффекты становятся понятнее, если использовать анимацию. Имеются два способа подготовки анимации. Первый основан на предварительном построении всех изображений (кадров) и последующем их проигрывании. Во втором способе кадры вычисляются в цикле и выводятся по мере готовности — старое изображение стирается, и рисуется новое.

В MATLAB имеются специальные средства для поддержки анимации с выводом изображения по мере подготовки данных. Однако если изображение сложное и число эпизодов невелико, то лучше подготовить кадры анимации заранее, а затем просто выводить картинки.

Кадры анимации подготавливаются при помощи графических функций, а чтобы воспроизвести эпизоды, нужны команды, представленные в табл. 14.14.

Таблица 14.14. Команды анимации

Имя	Назначение
moviein	Инициализация памяти для хранения матрицы кадров, нарисованных для текущих параметров осей
getframe	Создание отдельного кадра, помещаемого в столбец подготовленной матрицы
movie	Запуск анимации

Приведем простые примеры использования обоих способов. Для этого рассмотрим движение спектра трехдиагональной матрицы при изменении параметра. Зададим масштаб и подготовим матрицу:

```
» axis([-6 1 -6 6]), m=6; A1=diag([1:m]);
A2=diag(ones(m-1,1),-1);A3=diag(ones(m-1,1),1);
n=10; M=moviein(n); set(gca,'Nextplot','replacechildren')
```

Заметим, что обращение к примитиву вывода осей gca предупреждает функцию plot от перехода к стандартному режиму axis normal. Теперь приготовим кадры:

```
» for k=1:n,
ei=eig((k/n-1)*A1+A2-(k-1)*A3);
plot(real(ei),imag(ei),'ok'); M(:,k)=getframe;
end
```

Наконец, запустим анимацию, указав число повторов:

```
» movie(M,10)
```

Приведем на рис. 14.21 начальную и финальную картины этой анимации, использовав команду subplot, чтобы разместить на одном рисунке несколько изображений.

Рис. 14.21. Начальное и финальное расположение собственных значений

Тот же пример может быть реализован другим способом. Подготовим начальную картинку и получим дескриптор рисунка p:

```
» ei=eig(-A1+A2); axis([-6 1 -6 6]), hold on, n=10;
p=plot(real(ei),imag(ei),'.','EraseMode','none');
```

Указанный параметр 'EraseMode' позволяет дорисовывать новые детали — в нашем случае положения спектральных точек на графике. Теперь запустим цикл, в котором будем находить собственные числа, добавлять при помощи команды set новые точки к рисунку и сразу изображать их, выполняя команду drawnow:

```
» for i=1:5*n,
k=i/5; ei= eig((k/n-1)*A1+A2-(k-1)*A3);
set(p,'Xdata',real(ei),'Ydata',imag(ei)), drawnow
end
```

В результате выполнения цикла получается картина движения всех точек спектра при изменении параметра, см. рис. 14.22.

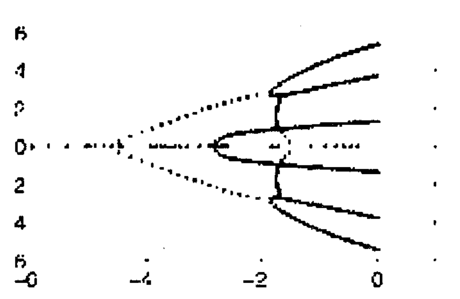


Рис. 14.22. Анимация движения собственных значений

В MATLAB имеются также команды оживления фазовых картин: двумерной — comet и трехмерной — comet3 (см. пример использования comet3 в разделе «Интегрирование дифференциальных уравнений» главы 15 «Численный анализ в MATLAB»). При записи в графические файлы результатов анимации или графиков, полученных при помощи команд comet и comet3, сохраняются данные последнего кадра или построения. Для фиксации анимационных картин следует перерисовать их при помощи обычных команд (plot и plot3). Именно таким образом был получен рис. 14.22.

Работа с изображениями

Рассмотренные ранее графические команды основаны на алгоритмах векторной графики, когда изображаемые объекты (кривые, поверхности, надписи) задаются координатами. В то же время многие изображения существуют в растровой форме, например фотографии. Система MATLAB имеет развитые средства для работы с растровыми объектами, включая подготовку растровых графических изображений, запись их в файл, считывание из файла картин, созданных другими программами. По умолчанию MATLAB работает с вещественными числами двойной точности (восемь байт для хранения числа типа double), а для работы с изображениями и сокращения требуемой памяти реализовано хранение данных также в виде однобайтных целых без знака (класс uint8).

Таблица 14.15. Команды работы с изображениями

Команда	Назначение
image	Вывод графического образа
imfinfo	Информация о графическом файле
imread	Чтение изображения из графического файла
imwrite	Запись изображения в графический файл

Команда image(C) выводит двумерный или трехмерный массив C как графический образ. Пусть размер массива есть M×N или M×N×3, тогда число M определяет количество прямоугольников по горизонтали, а N — по вертикали. Если C двумерный массив, то каждый элемент C рассматривается как значение индекса для массива, определяющего текущую палитру (команда colormap), и соответствующий этому

элементу C прямоугольник окрашивается в этот цвет. Этот способ задания изображения называется indexed image (индексированное изображение) В случае трехмерного массива C цвет точки (m,n) определяют элементы C(m,n,1:3), дающие соответственно доли красного, зеленого и синего цветов. При таком способе построения объекта image получаются изображения с числом цветов до 16 миллионов (truecolor image). В этом случае таблица цветов не используется.

Обращение image(X,Y,C), где X и Y — векторы, определяет размещение пиксела C(1,1) в точке с координатами (X(1),Y(1)) и пиксела C(M,N) соответственно в точке (X(end),Y(end)). По умолчанию MATLAB масштабирует выводимое изображение, поэтому пиксел обычно представляется в виде прямоугольника. Чтобы отменить масштабирование, нужно явно указать размеры. Например:

```
figure("Units",'pixels','Position',[100 100 N M])
```

Для записи растрового изображения (массив A) в файл FILE в графическом формате TYP применяется команда

```
imwrite(A,FILE,TYP)
```

При чтении и записи в качестве TYP выступают следующие графические форматы: **jpg (jpeg), tif (tiff), bmp, png, pcx, hdf, pcx, xmd**. Чтобы узнать тип изображения в файле, можно использовать команду imfinfo(FILE).

Команда считывания изображения из файла **FILE** в случае индексированного изображения имеет следующий вид:

```
[A,M]=imread(FILE,TYP)
```

или:

```
A=imread(FILE,TYP)
```

В массив A заносятся данные об изображении (цвета пикселов построчно), а массив M будет содержать таблицу цветов, если массив A двумерный (indexed image). Для считывания изображения truecolor image достаточно одного выходного параметра.

Приведем пример. Определим случайным образом массив:

```
» clear, B=64*rand(20,50);
```

и выведем изображение, назначив масштабирование, пропорциональное размерности массива, см. рис. 14.23:

```
» image(B); axis image
```

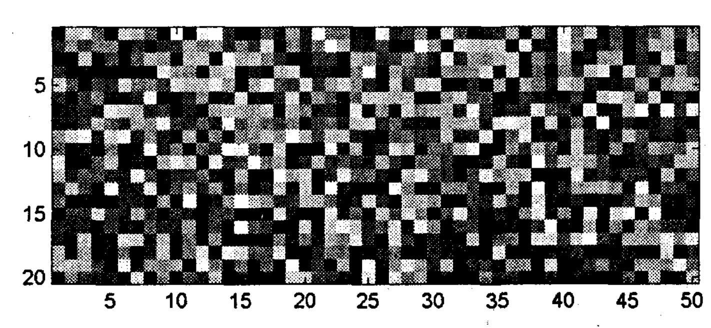


Рис. 14.23. Демонстрация команды image

Запишем данные с палитрой gray в формате bmp:

```
» imwrite(B,gray,'rnd.bmp','bmp')
```

Теперь считаем записанный файл при помощи команды

```
» [B1,M1]=imread("rnd.bmp",'bmp');
```

Выведем информацию о массивах B, B1 и M1:

```
» whos
  Name        Size            Bytes  Class
  B          20x50             8000  double array
  B1         20x50             1000  uint8 array
  M1        256x3              6144  double array
Grand total is 2768 elements using 15144 bytes
```

Видно, что данные об изображении записаны в экономном режиме (тип uint8 вместо double), а кроме того, сохранена палитра. Массив B1 получается из массива B в результате следующей операции:

```
» B1=uint8(B-1);
```

Тип uint8 не предназначен для арифметических операций. Поэтому если с изображением необходимо провести какие-нибудь численные действия, то вначале нужно преобразовать массив при помощи команды

```
» B2=double(B1)+1;
```

Заметим, что исходный массив B содержал вещественные числа, а массив B2 состоит из целых чисел.

В MATLAB для обработки изображений имеется пакет Image Proccesing Toolbox, см. главу 17 «Расширения MATLAB».

Интерактивная работа с графикой

Все графики, поверхности, специализированные изображения и анимации строятся в специальном графическом окне MATLAB. Это окно имеет свое меню, частично отличающееся от меню командного окна, а также набор значков для выполнения часто используемых операций.

Коротко опишем возможности оформления рисунков при помощи меню графического окна и контекстных меню, возникающих при указании мышью на графические объекты. Рассмотрим возможности версий MATLAB 5.3 и MATLAB 6.

Графическое окно MATLAB 5.3

Специфические команды графического меню версии MATLAB 5.3 связаны с пунктом Tools, см. табл. 14.16. Некоторые действия по оформлению рисунка можно выполнить при помощи команд меню или имеющихся значков. В частности, можно провести дополнительные линии и стрелки, нанести текст, повернуть изображение, изменить масштаб.

Таблица 14.16. Команды графического меню Tools

Имя	Назначение
Show Toolbar	Показ значков (переключатель)
Enable Plot Editing	Включение режима оформления (переключатель)
Axes Properties...	Вызов окна редактирования осей, см. рис. 14.27
Line Properties...	Вызов окна задания свойств кривой, см. рис. 14.26
Text Properties...	Вызов окна шрифтов
Unlock Axes Position	Фиксация размеров и положения рисунка в графическом окне (переключатель)
Show legend	Показ легенды (переключатель)
Add (Axes, Arrow, Line, Text)	Добавление элементов оформления (оси, стрелки, прямые, текст)
Zoom In	Увеличение масштаба
Zoom Out	Уменьшение масштаба
Rotate	Вращение рисунка

На рис. 14.24 представлено окно, появившееся в результате выполнения команды построения прямоугольника:

```
» box
```

где при помощи средств меню нанесены надписи, поясняющие назначение ряда значков, отличных от стандарта системы Windows.

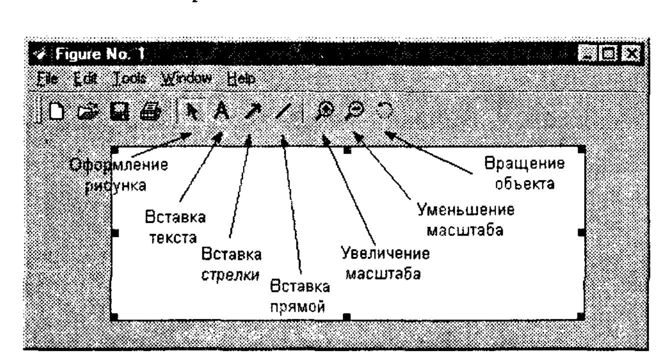


Рис. 14.24. Графическое окно

Для использования средств оформления графического окна выберем в меню Tools пункт Enable Plot Editing или используем соответствующий значок. После этого нужные размеры, цвет и тип линии можно определить при помощи пункта Line Properties из меню Tools или пункта Properties при нажатой правой кнопке мыши.

По команде Add появляется подменю, позволяющее выбрать один из следующих элементов оформления: изменение осей, нанесение стрелок и прямых, вставка текста. Указание начальной и конечных точек для рисования стрелок и линий, а также места для вставки текста производится при помощи мыши. Также при помощи мыши можно переместить нарисованные стрелки, линии, текст и изменить их раз-

меры. Чтобы удалить ненужный элемент, необходимо его выделить и нажать клавишу Del.

Выбор пункта графического меню Zoom In или соответствующего значка аналогичен действию команды zoom on. включающей режим масштабирования графика. Удерживая левую кнопку, можно выделить нужную прямоугольную область графика. Если нажать левую (правую) кнопку мыши возле нужной точки графика, то масштаб увеличивается (уменьшается) в два раза. Выбор пункта графического меню Zoom Out или соответствующго значка включает обратный режим масштабирования.

Для ряда действий удобно использовать возможности контекстных меню, появляющихся при нажатии правой кнопки мыши в области рисунка. Если графический курсор (стрелка) указывает на кривую, то при нажатии левой кнопки мыши данная кривая выделяется прямоугольными маркерами. Нажатие правой кнопки мыши приводит к появлению контекстного меню со следующими пунктами: Cut, Copy, Paste, Clear, Line Width, Line Style, Color..., Properties... . Например, на рис. 14.25 приведено окно, отвечающее процессу оформления рис. 14.2. Видно, что выделена кривая и появилось соответствующее контекстное меню.

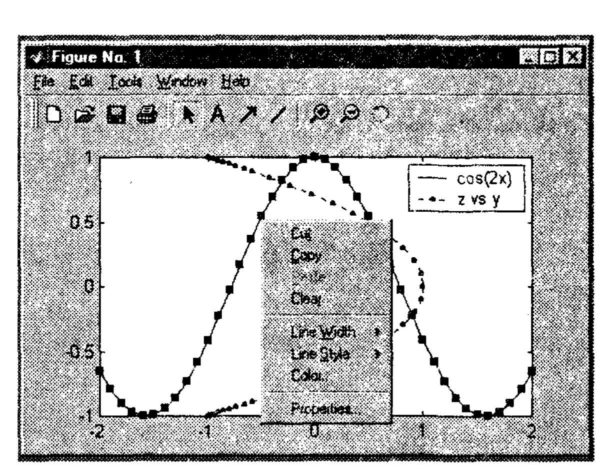


Рис. 14.25. Графическое окно с контекстным меню

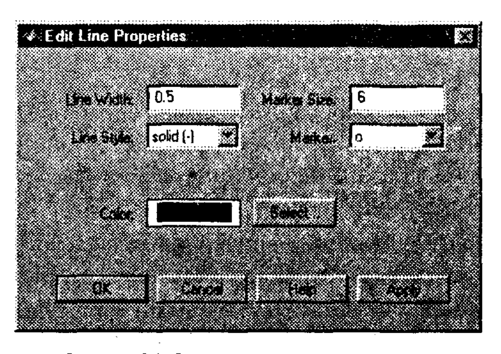


Рис. 14.26. Окно задания свойств кривой

При выборе пункта Properties... появляется окно Edit Line Properties, в котором можно определить толщину, тип и цвет кривой, задать тип и размер маркера, см. рис. 14.26.

При нажатии правой кнопки мыши при выделенном фрагменте текста появляется контекстное меню со следующими пунктами: Cut, Copy, Paste, Clear, String, Font Size, Font Style, Color..., Properties... Первые четыре пункта традиционны, пункт String позволяет отредактировать текст, следующие три пункта позволяют определить размер, тип шрифта и цвет, а последний пункт вызывает окно Edit Font Properties.

Если дважды нажать левую кнопку мыши в момент, когда графический курсор указывает на поле рисунка, не выделяя какого-либо объекта, то появится окно Edit Axes Properties, в котором можно изменить масштабирование, сопроводить рисунок заголовком, нанести сетку, изменить расстановку насечек, см. рис. 14.27.

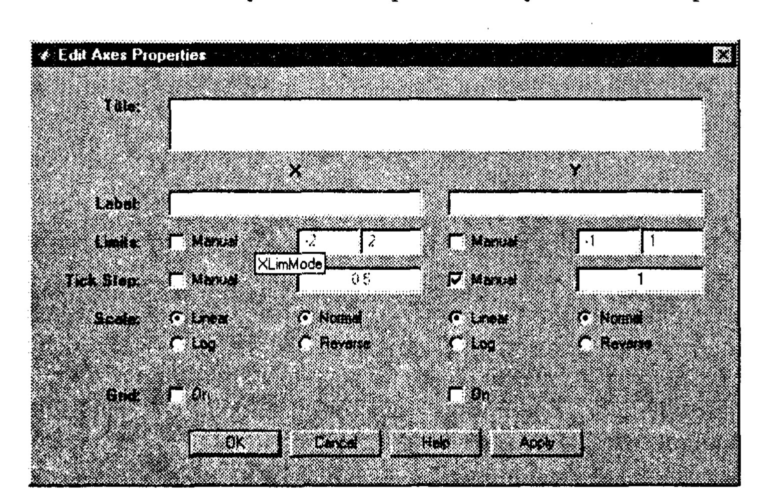


Рис. 14.27. Окно редактирования осей

Для изменения свойств графических объектов можно также использовать интерактивную среду редактора свойств propedit, входящего в состав системы графического интерфейса пользователя GUI.

Для демонстрации работы редактора propedit построим кривую (сам рисунок не приводится) и узнаем цвет рисунка:

```
» t=0:0.1:20; plot(t,t.*sin(t)), C1=get(gcf,'Color')
C1 =
   0.8000    0.8000    0.8000
```

Напомним, что три числа вектора C1 дают доли красного, зеленого и синего цветов (RGB). Вызовем редактор свойств прямо из командной строки:

```
» propedit(gcf)
```

В результате появится окно (см. рис. 14.28), в котором будут представлены свойства данного рисунка. Используя клавишу прокрутки, найдем свойство Color и укажем новые числа, чтобы изменить цвет.

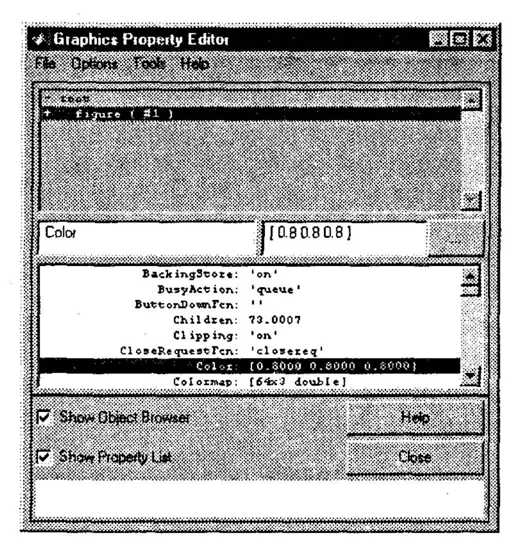


Рис. 14.28. Окно редактирования свойств графического окна

Графическое окно MATLAB 6

В версии MATLAB 6 возможности оформления рисунков средствами графического окна дополнены и частично изменены. В графическом меню окна теперь несколько пунктов предоставляют доступ к модификации рисунка. Так, в меню Edit содержатся пункты, позволяющие вызвать специальные окна для редактирования рисунка (Figure Properties...), осей (Axes Properties...), текущего графического объекта (Current Object Properties...).

В меню View находятся два пункта-переключателя — Figure Toolbar и Camera Toolbar, устанавливающие и убирающие соответствующие комплекты значков. Основной набор значков (Figure Toolbar) не изменился по сравнению с версией MATLAB 5.3, см. рис. 14.24, а описание режима работы с камерой и соответствующих значков в данной книге не рассматривается и за информацией следует обратиться к справке системы.

Новое меню Insert позволяет вставить элементы оформления: маркировки осей Xlabel, Ylabel и Zlabel, текстовую легенду Legend и цветовую шкалу Colorbar, добавить стрелку Arrow, прямую Line и текст Text, а также установить дополнительные оси Axes и источник света Light.

Меню Tools состоит из следующих пунктов: включение режима редактирования Edit Plot, установка режима увеличения Zoom In и уменьшения Zoom Out масштаба, вращение изображения Rotate 3D, перемещение камеры Move Camera и дополнительные параметры Camera Motion, Camera Axis, Camera Reset, обработка данных Data Statistics и Basic Fitting.

В версии MATLAB 6 усовершенствованы окна, вызываемые для редактирования осей, линий, см., например, рис. 14.29.

Добавились новые возможности по обработке изображенных данных. Теперь непосредственно из меню Tools (пункт Data Statistics) можно вызвать окно статистики с рядом характеристик исходных данных. Это позволяет быстро посмотреть информацию о минимальном и максимальном значениях, медиане, среднем и других параметрах, а также нанести на график нужную характеристику, для чего достаточно пометить соответствующий пункт таблицы, см. рис. 14.30.

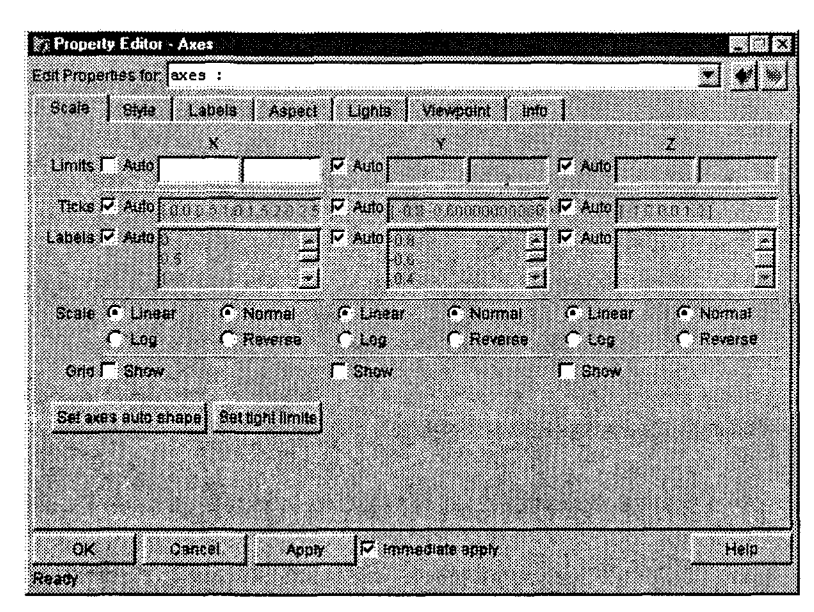


Рис. 14.29. Редактор свойств

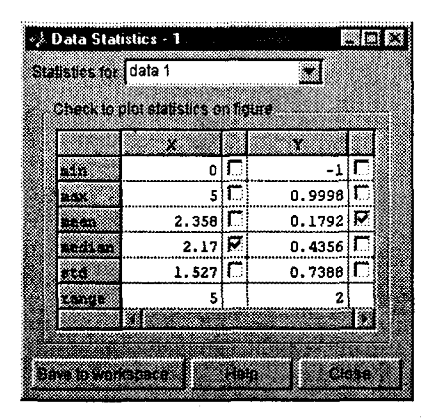


Рис. 14.30. Окно статистики

Изображенные данные можно дополнить построениями интерполяционных полиномов разного порядка и т. п. при помощи специального окна Basic Fitting, см. рис. 14.31.

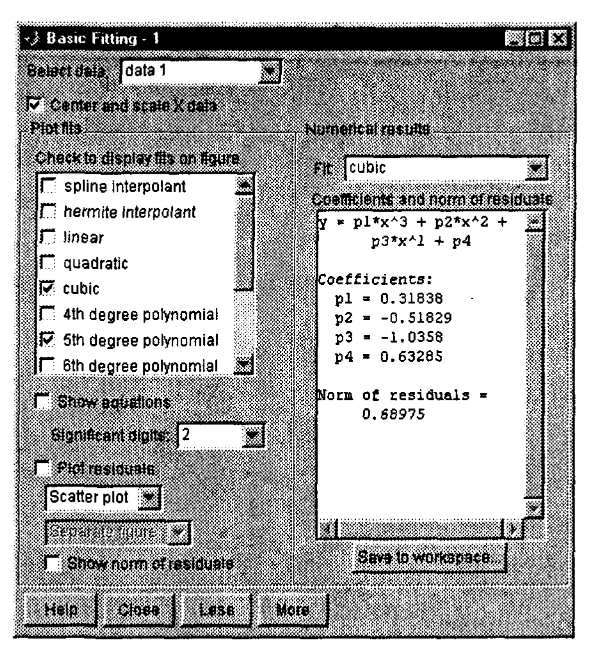


Рис. 14.31. Дополнительные возможности интерполяции для обработки данных

Печать и запись рисунков в файл

Команды печати в MATLAB служат для получения иллюстраций (твердой копии изображения) и подготовки графических файлов, которые в дальнейшем можно использовать в различных системах редактирования и размещать в Интернете. В MATLAB имеется широкий набор графических форматов. При помощи команды печати или меню графического окна можно распечатать изображение, записать рисунок в собственном формате (fig) и сохранить его в нескольких графических форматах.

Таблица 14.17. Команды печати

Имя	Назначение
`print`	Сохранение рисунка в файле
`printout`	Задание параметров команды `print`
`prtsc`	Печать рисунка

Команда печати обычно имеет дополнительные параметры, указывающие графический формат (табл. 14.18) и некоторые особенности. Так, для добавления рисунка к существующему файлу достаточно набрать

```
print -append file
```

В MATLAB используются также другие форматы, полную информацию о которых можно получить, обратившись к справке help print. Кроме того, сохранить изображение в каком-либо формате можно при помощи пункта Export... из меню File.

Таблица 14.18. Графические форматы

Параметр	Назначение
-dwin	Черно-белая печать
-dwinc	Цветная печать
-dmeta	Сохранение в буфере в формате emf
-dbitmap	Сохранение в буфере в формате bmp
-dps	Черно-белая печать (рисунок) в стандарте PostScript уровня 1
-dpsc	Цветная печать в стандарте PostScript уровня 1
-dps2	Черно-белая печать в стандарте PostScript уровня 2
-dpsc2	Цветная печать в стандарте PostScript уровня 2
-deps	Черно-белая печать в стандарте Encapsulated PostScript уровня 1
-depsc	Цветная печать в стандарте Encapsulated PostScript уровня 1
-deps2	Черно-белая печать в стандарте Encapsulated PostScript уровня 2
depsc2	Цветная печать в стандарте Encapsulated PostScript уровня 2
-dill	Графический формат Adobe Illustrator 88
-djpeg<nn>	JPEG-формат, качество определяется параметром nn (по умолчанию nn равно 75)
-dtiff	Графический формат TIFF

Назначения можно изменить прямо в команде print, например для записи в формате eps второго рисунка (Figure No. 2) в файл abc достаточно выполнить следующую команду:

```
print -deps -f2 abc
```

Параметры страницы можно определить при помощи команды orient TYPE с параметром TYPE, принимающим значения portrait (портрет), landscape (ландшафт), tall (печать во всю страницу), а также явно указать графическое окно и параметр orient(N,TYPE). Того же самого можно добиться при помощи специального окна Page Setup, вызываемого при выборе одноименного пункта меню File графического окна, см. рис. 14.32.

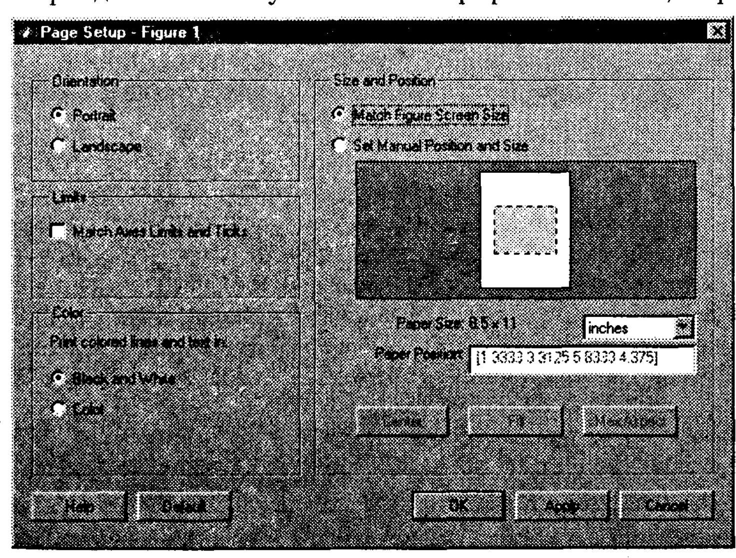


Рис. 14.32. Окно определения параметров страницы в MATLAB 5.3

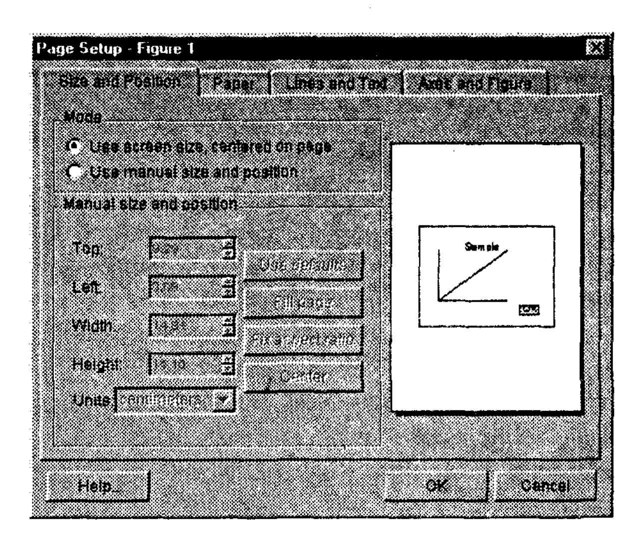


Рис. 14.33. Окно определения параметров страницы в MATLAB 6

Из всех возможностей данного окна отметим пункт Match Figure Screen Size, позволяющий увязать величину графического окна на экране дисплея с площадью картинки на странице или размером изображения в графическом файле. В MATLAB 6 окно для определения параметров страницы Page Setup приобрело другой вид, см. рис. 14.33.



Численный анализ в MATLAB

До этой главы вычислительные возможности MATLAB были представлены лишь для задач линейной алгебры. Теперь рассмотрим команды для решения нелинейных задач, вычисления интегралов и интерполяции, интегрирования дифференциальных уравнений и т. п.

Интересно отметить, что разработчиком первой версии системы MATLAB был Клиффорд Моулер — известный специалист по численному анализу, ряд книг которого переведен на русский язык [53, 71, 72]. Для успешного применения функций MATLAB к решению разнообразных задач можно рекомендовать знакомство с упомянутыми книгами, а также с книгами отечественных авторов по численному анализу, где приводятся теория и практические рекомендации по применению тех или иных вычислительных процедур [44, 45, 49, 52, 68].

Алгоритмы численного решения постоянно совершенствуются, и функции MATLAB также модернизируются, чтобы лучше и эффективнее справляться с задачами повышающейся сложности. Следует помнить, что опасно доверять численному результату, полученному одним методом, и здоровый скептицизм дополнительных проверок всегда уместен в серьезных делах. Поэтому на ряде примеров мы постараемся показать, какие неожиданные решения могут получиться для вроде бы стандартных задач.

Функции для вычисления интегралов, решения задач минимизации, интегрирования дифференциальных уравнений находятся в каталоге toolbox\matlab\dunfun, и их список можно получить по команде `help funfun`. Команды работы с конечными разностями, функции для вычисления быстрого преобразования Фурье и фильтрации содержатся в директории toolbox\matlab\datafun. Функции для работы с полиномами, интерполирования, геометрического анализа и т. д. находятся в каталоге toolbox\matlab\polyfun.

Работа с полиномами

В MATLAB имеется мощный набор команд для работы с полиномами, обеспечивающий их перемножение, дифференцирование, вычисление корней и т. д. Ряд

функций позволяет работать также с дробно-рациональными выражениями, числители и знаменатели которых есть полиномы. Полином в MATLAB задается вектором коэффициентов начиная с коэффициента при старшей степени. Например, кубическому полиному $x^3+12x-9$ соответствует вектор

```
» pn=[1 0 12 -9]
pn =
    1     0    12    -9
```

Команды работы с полиномами представлены в табл. 15.1, где использованы следующие обозначения: a, b, d, p, q, — полиномы, r — вектор-столбец корней, S — массив точек, M — квадратная матрица. Выходные параметры указаны только в том случае, если их несколько.

Таблица 15.1. Команды работы с полиномами

Имя	Назначение
poly(r)	Задание полинома по его корням
polyval(p,S)	Вычисление значений полинома p в точках S
polyvalm(p,M)	Вычисление значений полинома p для матричного аргумента M
polyder(p)	Дифференцирование полинома p
polyder(a,b)	Дифференцирование произведения полиномов a и b
[d,p]=polyder(b,a)	Дифференцирование дроби с числителем b и знаменателем a
conv(a,b)	Умножение полиномов
[q,p]=deconv(a,b)	Деление полиномов (получение частного и остатка)
[d,p,q]=residue(b,a)	Разложение дроби с числителем b и знаменателем a на простые дроби (d — вычеты, p — полюс, q — частное)
roots(p)	Вычисление корней полинома

Команда polyval вычисляет значения полинома для элементов массива S, рассматривая их как точки, а команда polyvalm работает с матричным аргументом M. Поясним это следующим примером. Определим линейный полином и матрицу второго порядка:

```
» p=[1 1]; M=[1 0; 1 2];
```

Обратимся к функции polyval, рассматривая M как массив точек:

```
» polyval(p,M)
ans =
     2     1
     2     3
```

Теперь вычислим полином от матрицы:

```
» polyvalm(p,M)
ans =
     2     0
     1     3
```

Приведем примеры использования операций деления (команда deconv) и вычисления производной (polyder).

Разделим полином p=(x-1)7 на x2+1:

```
» [p1,p2]=deconv(poly(ones(7,1)),[1 0 1])
p1 =
     1    -7    20   -28    15     7
p2 =
     0     0     0     0     0     0    -8    -8
```

Размерность вектора, в котором даются коэффициенты остатка, совпадает с размерностью делимого, а степень полученного в остатке полинома определяется первым ненулевым коэффициентом. Для приведенного примера остаток есть -8x-8.

Вычислим производную для полинома p2:

```
» polyder(p2)
ans =
    -8
```

Если известны корни полинома, то его можно задать при помощи команды poly.

Хотя нахождение корней полинома есть стандартная задача численного анализа и обычно не возникает проблем с вычислениями, существуют ситуации, когда полученные решения могут оказаться далеки от реальных значений корней. В [72] анализируется пример Уилкинсона (1963), когда при нахождении корней полинома двадцатого порядка

$$\prod_{k=1}^{20}(x-k)$$

возникают комплексные решения, хотя все корни исходного полинома вещественные. В современной реализации MATLAB для этого теста получаются довольно хорошие результаты. Определим полином двадцатого порядка и вычислим корни при помощи функции roots:

```
» p0=poly(1:20)'; roots(p0)
```

Хотя полученные решения и отличаются от точных, все они вещественные. Наибольшая погрешность в полпроцента получается для четырнадцатого и пятнадцатого корня:

```
1.0000  2.0000  3.0000  4.0000  5.0000  6.0000  7.0000
7.9999  9.0005  9.9972  11.0107  11.9725  13.0565  13.9220
15.0842  15.9319  17.0360  17.9857  19.0032  19.9997
```

В данном случае получился неплохой результат, однако при некотором повышении степени полинома вновь могут появиться комплексные решения. Определим полином двадцать четвертого порядка, вычислим его корни и покажем их распределение на рис. 15.1. Видно, что первые девять корней воспроизводятся хорошо. При вычислении следующих корней погрешность увеличивается и вместо вещественных чисел получаются комплексные:

```
» pp=poly(1:24)'; roo=roots(pp); plot(roo,'+')
```

Полином может быть получен в результате перемножения двух полиномов. Далее дан пример получения полинома с кратным корнем:

```
» p=[1]; for n=1:7, p=conv(p,[1 -1]); end;
```

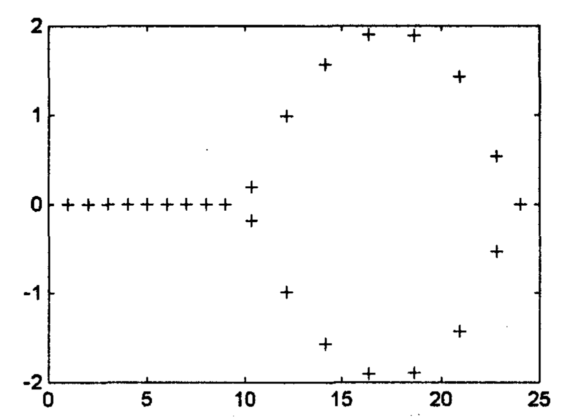


Рис. 15.1. Комплексные решения для полинома с вещественными корнями

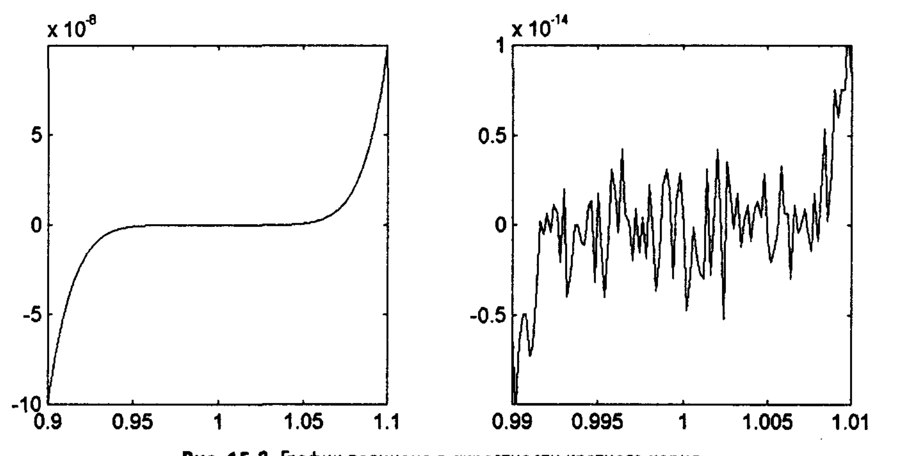


Рис. 15.2. График полинома в окрестности кратного корня

Чтобы нарисовать график полинома, нужно задать вектор значений независимой переменной, вычислить значения полинома в этих точках и обратиться к команде plot. Построим график полинома, выбрав небольшой отрезок в окрестности корня (точка x=1). Команда определения масштаба axis позволяет точно указать максимум и минимум, если использовать константу Inf:

```
» x=0.9:.0002:1.1; y=polyval(p,x);
subplot(121), plot(x,y); axis([.9 1.1 -Inf Inf])
subplot(122), plot(x,y); axis([.99 1.01 -1e-14 1e-14])
```

На рис. 15.2 график слева выглядит так, как предсказывает теория для гладкой функции с единственным корнем: полином положителен при x>0, отрицателен при x<0 и равен нулю только при x=0. На правом рисунке из-за ошибок округления получается график с большим числом нулей, которое увеличивается, если взять большее число точек для построения графика.

Найдем корни полинома и изобразим их на рис. 15.3. Вместо семи одинаковых корней MATLAB выводит набор комплексных чисел, сгруппированных около точного значения, равного единице:

```
» xp=roots(p), plot(xp,'o'), grid
xp =
   1.0103
   1.0064 + 0.0081i
   1.0064 - 0.0081i
   0.9977 + 0.0100i
   0.9977 - 0.0100i
   0.9908 + 0.0044i
   0.9908 - 0.0044i
```

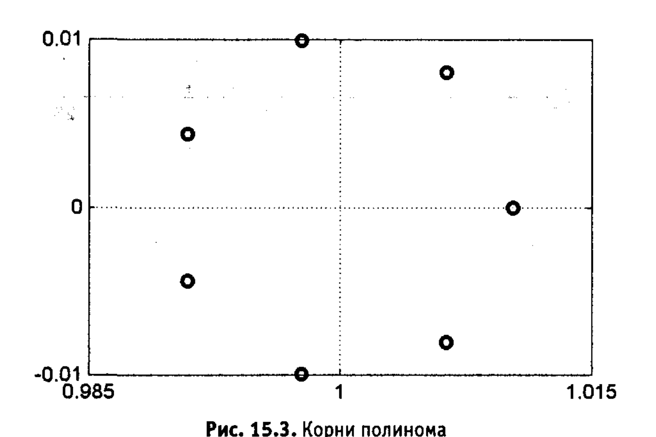


Рис. 15.3. Корни полинома

Приведенные примеры показывают, какого рода особенности могут встретиться в расчетах. При нахождении корней полиномов с чисто вещественными корнями получаются комплексные числа, а вычисление значений полинома с кратным корнем дает стохастическую картину в окрестности этого корня.

Решение уравнений и минимизация

Полный вариант обращения к команде поиска нулей функции одной переменной fzero имеет вид:

```
[z,fz,exi,opt]=fzero("FUN",x0,OPT,P1,P2,...)
```

Здесь FUN — имя функции, x0 — начальное приближение, OPT — структура, определяющая параметры метода. Узнать и изменить значения полей структуры можно при помощи команды optimset. Запуск optimset('fzero') информирует о значениях, принятых по умолчанию. Если функция FUN зависит от параметров P1, P2, ... (не более десяти), то их можно передавать, указывая после OPT. При обращении к команде fzero имя функции FUN должно быть указано обязательно, а все остальные параметры или некоторые из них могут быть опущены. Выходными параметрами являются: решение уравнения z, значение функции на решении fz, переменная успешности расчета (флаг) exi (1) и параметры opt. Число выходных параметров может быть сокращено до одного, в этом случае таким параметром является решение уравнения.

Приведем пример обращения к команде fzero с полным набором выходных параметров. Определим функцию при помощи конструкции inline:

```
» [x,f,exi,opt]=fzero(inline("x^3-1"),2)
Zero found in the interval: [0.72, 2.9051].
x =
     1
f =
     0
exi =
     1
opt =
    iterations: 28
     funcCount: 28
     algorithm: "bisection, interpolation"
```

Действует также старый формат обращения к команде fzero:

```
fzero("FUN",x0,tol,trace,P1,P2)
```

Здесь tol — точность, а ненулевое значение trace указывает, что в процессе решения следует получать промежуточные результаты.

Для нахождения корней системы нелинейных уравнений можно использовать команду solve из пакета Symbolic Math Toolbox, см. главу 17 «Расширения MATLAB» или реализовать алгоритмы численного решения уравнений самостоятельно, пример использования метода Ньютона дан в главе 18 «Дополнения и примеры».

Команды для нахождения минимума функции одной или нескольких переменных приведены в табл. 15.2.

Таблица 15.2. Команды минимизации

Имя	Назначение
fmin	Минимизация функции одной переменной
fmins	Минимизация функции нескольких переменных

Обращение к команде минимизации функции одной переменной имеет вид:

```
[xmin,opt]=fmin("FUN",A,B,OPT,P1,P2,...)
```

Здесь FUN — имя функции, [A,B] — заданный интервал, а параметры минимизации (погрешность, максимальное число итераций и другие) передаются вектором из восемнадцати элементов OPT. Назначение элементов (параметров) OPT описано в табл. 15.3. Имя функции при обращении должно быть указано обязательно, а остальные параметры могут отсутствовать. Если указан один выходной параметр, то возвращается xmin. Если функция FUN зависит от параметров P1, P2, ... (не более десяти), то их можно передавать, указывая после OPT. При передаче параметров P1, P2, ... структуры OPT они должны присутствовать в списке хотя бы в виде пустой переменной [].

Результатом выполнения команды fmin будет значение минимума и вектор управляющих параметров opt, которые использовались алгоритмом. В частности, параметр opt(8) дает вычисленное в точке минимума значение функции, а параметр

opt(10) содержит число произведенных итераций, см. табл. 15.3. Вектор исходных назначений параметров для функций fmin и fmins возвращает команда

```
OPT=foptions
```

Для изменения параметров следует присвоить соответствующему элементу OPT требуемое число, например для получения промежуточных результатов достаточно ввести OPT(1)=1, а затем обратиться к команде минимизации, указав в списке параметров измененный вектор OPT.

Поиск минимума функции нескольких переменных осуществляется по команде

```
[xmin,opt]=fmins("FUN",x0,OPTI,[],P1,...,P10)
```

Здесь разыскивается локальный минимум в окрестности точки, задаваемой вектором x0, в остальном описание параметров аналогично рассмотренному для команды fmin.

Таблица 15.3. Параметры команд минимизации

Номер параметра	Описание	Значение по умолчанию
1	Вывод промежуточных результатов (да — 1, нет — 0)	0
2	Погрешность для аргумента	1e-4
3	Погрешность для функции	1e-4
4	Погрешность для ограничений	1e-6
5	Стратегия	0
6	Оптимизация	0
7	Алгоритм линейного поиска	0
8	Значение целевой функции	0
9	Управление градиентом (да — 1, нет — 0)	0
10	Количество итераций	0
11	Количество вычислений градиента	0
12	Количество вычислений ограничений	0
13	Число ограничений	0
14	Максимальное число итераций	500 для fmin 100n для fmins
15	Наличие целевой функции (да — 1, нет —0)	0
16	Минимальный шаг при вычислении градиента	1e-8
17	Максимальный шаг при вычислении градиента	0.1
18	Размер шага	1

Рассмотрим пример минимизации функции одной векторной переменной и двух векторных параметров. Подготовим функцию в файле fu.m:

```
function out=fu(z,xx,yy)
% z - point,  xx, yy - parameter vectors
out=-sum(1./((z(1)-xx).^2+(z(2)-yy).^2+.1));
```

Зададим векторы xx и yy:

```
» n=3; xx=rand(n,1)', yy=rand(n,1)'
```

```
xx =
   0.7095    0.4289    0.3046
yy =
   0.1897    0.1934    0.6822
```

Обратимся к команде минимизации функции нескольких переменных fmins и выведем найденное значение минимума:

```
» [zmin,opt]=fmins("fu",[.5 .5],[],[],xx,yy); zmin
zmin =
   0.5344    0.2165
```

В восьмом элементе массива параметров opt должно содержаться значение функции в точке минимума:

```
» opt(8)
ans =
  -19.2724
```

Проверим, что значение функции fu в найденной точке минимума zmin действительно совпадает с opt(8):

```
» fu(zmin,xx,yy)
ans =
  -19.2724
```

Численное интегрирование и дифференцирование

Вычисление интегралов и производных во многих случаях довольно рутинная операция при использовании пакетов аналитических вычислений. В MATLAB имеется пакет Symbolic Math Toolbox, который предназначен для проведения аналитических операций. Вместе с тем часто задача содержит числовые данные или интеграл не берется в квадратурах, поэтому для работы нужны эффективные вычислительные процедуры. Для получения информации по функциям следует обратиться к справке

```
help funfun
```

Таблица 15.4. Функции численного интегрирования

Команда	Действие
trapz	Метод трапеций
quad	Метод Симпсона
quad8	Метод Ньютона–Котеса (формулы 8 порядка)
dblquad	Интеграл по области

Функция trapz вычисляет значение одномерного интеграла методом трапеций, если заданы абсциссы и ординаты или только ординаты (в этом случае размер шага считается равным единице). Посчитаем интеграл двумя способами.

Укажем x и y:

```
» x=-1:0.1:1; y=cos(x); trapz(x,y),
ans =
    1.6815
```

А теперь опустим x и результат умножим на величину шага интегрирования (отношение величины интервала к числу ячеек):

```
» trapz(y)*(x(end)-x(1))/(length(x)-1)
ans =
    1.6815
```

Команды quad и quad8 предназначены для интегрирования функций и используют рекурсию, причем глубина рекурсии ограничена уровнем 10, чтобы избежать бесконечного дробления интервала. Обращение к команде quad для интегрирования функции F (m-файл или встроенная функция) на интервале [a,b] с заданной относительной точностью tol имеет вид:

```
[s,n]=quad("F",a,b,tol,trace)
```

Здесь s — значение интеграла и n — число вычислений подынтегральной функции. Точность вычисления интеграла tol по умолчанию равна 1е-3. Параметры tol и trace могут быть опущены, а если параметр trace имеет ненулевое значение, то процесс вычисления интеграла будут проиллюстрирован графически. Обращение к функции quad8 аналогично рассмотренному для quad. Для обеих функций вместо скалярной точности tol можно задать вектор [Rtol Atol], определяющий относительную и абсолютную погрешности. Если интегрируемая функция зависит от параметров P1, ..., PN, то указывать их при вызове quad или quad8 следует после переменной trace, причем сами параметры tol и trace могут присутствовать фиктивно, то есть:

```
[s,n]=quad("F",a,b,[],[],P1,...,PN)
```

В завершение приведем простой пример:

```
» [s,n]=quad("cos",-1,1,1e-6,1)
s =
    1.6829
n =
    129
```

Мы не приводим графики распределения точек, полученные в процессе интегрирования, только заметим, что для вычисления этого простого (для систем аналитических вычислений) интеграла потребовалось достаточно много итераций. Для интегрирования при помощи функции quad8 оказалось достаточным 33 точек. С появлением в MATLAB пакета Symbolic Math Toolbox на основе символьного ядра Maple стало возможным использовать аналитическое вычисление интегралов, см. главу 17 «Расширения MATLAB».

Теперь перейдем к операциям численного дифференцирования. Вычисление конечных разностей представлено командами, перечисленными в табл. 15.5.

Для одномерного массива y вызов функции diff(y,1) или diff(y) порождает массив с числом элементов, меньшим на единицу. Для двумерного массива разности вычисляются по столбцам. Функция diff(y,n) находит конечные разности поряд-

ка n. По заданным значениям аргумента x и функции y производную порядка n можно оценить при помощи отношения diff(y,n)/diff(x,n).

Таблица 15.5. Вычисление конечных разностей

Команда	Назначение
diff(y,n)	Вычисление конечных разностей n-порядка для массива y
[FX,FY]=gradient(F,HX,HY)	Вычисление градиента функции двух переменных, заданной массивом F
del2(U,HX,HY)	Вычисление разностного оператора Лапласа для двумерного массива U

В качестве примера использования команды **diff** проведем проверку массива точек на монотонность. Напомним, что команда all выведет 1 (истина), если условие выполнено для всех элементов, и 0 (ложь) в противном случае:

```
» v=rand(4,1)', all(diff(v)>0)
v =
    0.3784    0.8600    0.8537    0.5936
ans =
     0
```

Градиент функции n переменных, заданной значениями в виде n-мерного массива F, вычисляется при помощи команды

```
[F1,...,Fn]=gradient(F,H1,...,Hn)
```

Необязательные параметры H1, ..., Hn служат для определения шагов сетки по координатам. В случае равномерной сетки это сами шаги (скалярные величины), а для неравномерной сетки параметры H1, ..., Hn задают одномерные массивы координат. При обращении gradient(F) шаги сетки считаются равными единице, а вызов функции gradient(F,H) с единственным скалярным параметром H соответствует вычислению градиента с одинаковым шагом по всем координатам.

Функция del2 вычисляет одну четвертую разностного оператора Лапласа от заданной таблицей функции при помощи аппроксимации второго порядка:

```
del2(U,H1,...,Hn)
```

Назначения параметров H1, ..., Hn аналогичны рассмотренным ранее для функции gradient, а выходной массив имеет ту же размерность, что и массив U.

Приведем пример вычисления конечных разностей для квадратичной функции xy (седловая функция). Введем равномерную сетку по координате x и неравномерную — по y:

```
» n=10; x=-1:1./n:1; y=sin(pi/2*x);
```

Подготовим массив значений функции:

```
» [X,Y]=meshgrid(x,y); Z=X.*Y;
```

Вычислим лапласиан и убедимся, что полученный массив состоит из элементов, близких к значению константы погрешности **eps**, это не удивительно при двукратном численном дифференцировании данной функции:

```
» DZ=del2(Z,x,y); max(max(DZ)), min(min(DZ))
ans =
  2.1301e-014
ans =
 -2.1301e-014
```

Построим линии уровня исходной функции и векторное поле градиента функции, см. рис. 15.4. Используем для этого графические команды subplot, contour и quiver (поле направлений):

```
» subplot (121), contour(x,y,Z),
subplot(122), [d1,d2]=gradient(Z); quiver(x,y,d1,d2)
```

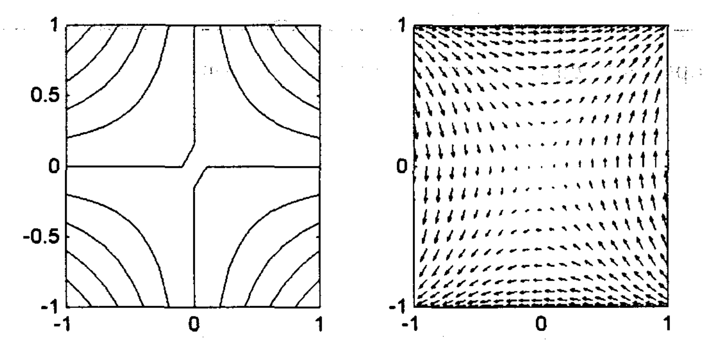


Рис. 15.4. Линии равного уровня (слева) и градиент (справа) седловой функции

Отметим, что команда contour построила линию уровня с нулевым значением, обойдя точку x=y=0.

Интерполяция и приближение функций

Интерполяция позволяет оценить значение заданной таблично функции в тех точках, для которых данных нет. Полиномиальная интерполяция применяется для работы с произвольными данными, а интерполяция, основанная на быстром преобразовании Фурье (БПФ), — для периодических наборов данных. Для обработки экспериментальных данных и в некоторых других случаях нужны команды, позволяющие аппроксимировать функции методом наименьших квадратов.

В табл. 15.6 перечислены команды интерполяции. При их описании использованы следующие обозначения: X, Y, Z — заданные массивы (точки и значения функции), Xi и Yi — массивы точек, в которых должны быть вычислены значения интерполяционного полинома, M — описание метода интерполирования (линейная, кубическая или сплайн-интерполяция).

Для проведения одномерной интерполяции по данным, организованным в таблицу из точек X и значений функций Z, используется команда

```
Zi=interp1(X,Z,Xi,'METHOD')
```

Значения интерполируемой функции будут определены в точках Xi, и выходной массив Zi будет того же размера, что и массив Xi. Дополнительный параметр METHOD задает метод интерполирования: по ближайшей точке (nearest), линейная интер-

поляция (linear), которая действует по умолчанию, кубическая (cubic) или интерполяция кубическим сплайном (spline). Для всех методов величины элементов X должны меняться монотонно. Для узлов Xi, лежащих вне интервала, определенного X, соответствующим значениям выходного массива будут присвоены нечисловые константы NaN. Если параметр X отсутствует, то считается, что значения функции вычислены на равномерной целочисленной сетке X=1:N, где N=size(Z,1).

Таблица 15.6. Команды интерполяции

Команда	Описание
interp1(X,Z,Xi,'M')	Одномерная интерполяция
interp2(X,Y,Z,Xi,Yi,'M')	Двумерная интерполяция
interp3(X,...,Z,Xi,...,'M')	Трехмерная интерполяция (3 массива координат)
interpn(X,...,Z,Xi,...,'M')	n-мерная интерполяция (n массивов координат)
Interpft(Z,N)	Интерполяция периодической функции
griddata(X,Y,Z,Xi,Yi)	Интерполяция на неравномерной сетке
spline(X,Z,Xi)	Интерполяция кубическим сплайном
polyfit(X,Z,N)	Определение методом наименьших квадратов полинома заданной степени N, аппроксимирующего таблицу чисел

Приведем простой пример. Интерполируем функцию синуса на промежутке [0,2] и вычислим значения для точек 0, 1, 2, 3. Для последней точки числового значения нет, поскольку точка x=3 вне интервала, на котором задана таблица данных:

```
» x=0:.5:2; zi=interp1(x,sin(x),[0:3],'cubic')
zi =
        0    0.8415    0.9093       NaN
```

Многомерный случай отличается от одномерной интерполяции большим числом массивов для задания координат. Рассмотрим команду n-мерной интерполяции:

```
interpn(X1,...,Xn,Z,Xi1,...,Xin,'METHOD')
```

В качестве параметра METHOD используются те же имена, что и для одномерной интерполяции: linear, cubic, nearest и spline. Каждая n-мерная матрица из списка X1, ..., Xn определяет значения одной координаты функции n переменных, представленной массивом значений Z. Координаты X1, ..., Xn должны изменяться монотонно, причем в случае равномерного расположения узлов для ускорения интерполирования можно указывать в качестве параметра METHOD значения *linear, *cubic, *nearest. Параметр со звездочкой означает, что вычисления ведутся по формулам с постоянным шагом, что и приводит к ускорению.

Для двумерных матриц и массивов большей размерности имеется возможность быстро получить расширенный массив данных, вычислив дополнительные значения для новых узлов, располагающихся между старыми узлами. При вызове interp2(Z,1), interp3(Z,1) или interpn(Z,1) массив значений будет расширен по каждой размерности за счет вставки промежуточных интерполированных точек. При вызове interp2(Z,N), interp3(Z,N) или interpn(Z,N) подобное расширение рекуррентно будет проведено N раз.

Продемонстрируем действие данной команды для двумерного массива 3×3:

```
» Z
Z =
     1     3     5
     2     5     8
     0     2     4
```

В результате выполнения команды при N=1 получаем расширенную матрицу размером 5×5:

```
» interp2(Z,1,'cubic')
ans =
    1.0000    2.0000    3.0000    4.0000    5.0000
    1.8750    3.2500    4.6250    6.0000    7.3750
    2.0000    3.5000    5.0000    6.5000    8.0000
    1.3750    2.7500    4.1250    5.5000    6.8750
         0    1.0000    2.0000    3.0000    4.0000
```

Не приводя сам массив, укажем размерность результата для той же исходной матрицы при вызове

```
» size(interp2(Z,3,'cubic'))
ans =
    17    17
```

При многомерной интерполяции массивы точек создаются при помощи команды ndgrid (аналог двумерной команды meshgrid). Время, необходимое для интерполяции, зависит от длины массива. Задавая в качестве дополнительного параметра обрабатываемое число точек, можно ускорить процесс обработки данных. Например, можно указать число, являющееся степенью двойки или просто выбирая число, меньшее числа точек. Если указывается число, превосходящее реально имеющееся, то массивы будут пополнены нулями.

Функция griddata применяется для двумерной интерполяции данных, определенных в точках, координаты которых даются векторами X и Y, и получения массива значений Zi для упорядоченных массивов координат Xi, Yi. Отметим, что в этом случае значения не обязательно упорядочены:

```
Zi=griddata(X,Y,Z,Xi,Yi)
```

Приведем пример. Зададим поверхность сорока точками, вычисленными случайным образом:

```
» x=2*rand(40,1)-1; y=2*rand(40,1)-1; z=-sin(x.*y);
```

Построим равномерную сетку и интерполируем в ее узлах данные:

```
» t=-1:.1:1; [Xi,Yi]=meshgrid(t); Z=griddata(x,y,z,Xi,Yi);
```

Построим полученную интерполированием поверхность и нанесем исходные точки:

```
» mesh(Xi,Yi,Z), hold("on"), plot3(x,y,z,'ok')
```

Обратим внимание, что поверхность на рис. 15.5 построена не для всех точек, определенных массивами Xi, Yi. Это связано с тем, что соответствующие элементы мас-

сива имеют нечисловой характер (NaN). Действительно, выведем несколько элементов второй строки:

```
» Z(2,1:5)
ans =
      NaN      NaN   -0.6583   -0.6025      NaN
```

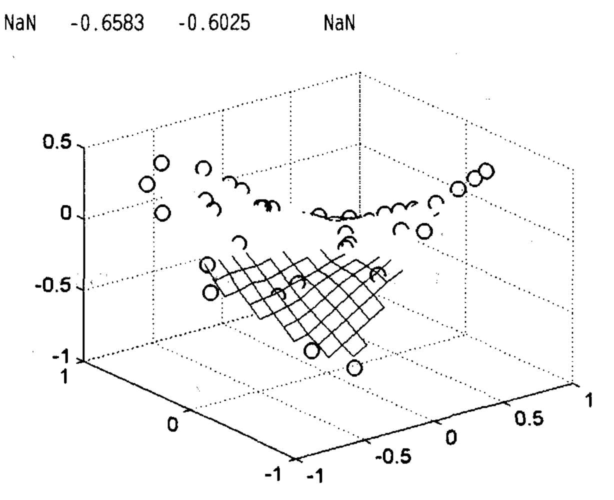


Рис. 15.5. Набор случайных точек и результат действия команды griddata

Команда spline(X,Y,Xi) выведет значения кубического сплайна для узлов Xi по таблице (X,Y) аналогично тому, что делает команда interp1(X,Y,Xi,'spline'). Кроме того, вариант этой команды PP=spline(X,Yi) позволяет получить сам сплайн (так называемая ppform форма) и в дальнейшем вычислять его значения при помощи функции ppval. Кроме того, в состав мини-пакета по работе со сплайн-интерполяцией входят команды mkpp и unmkpp, которые используются для формирования кусочно-гладкого полинома и вывода его характеристик.

Рассмотрим аппроксимацию функции sin(x^2). Подготовим довольно редкую сетку и построим сплайн. Вычислим значения сплайна и на рис. 15.6 приведем графики сплайна (сплошная кривая) и исходной функции (пунктир), а также набор точек, по которым вычислялся сплайн (кружки):

```
» x=0:.5:5; y=sin(x.^2); xx=0:.05:5; yy=spline(x,y,xx);
plot(x,y,'ok',xx,yy,'k',xx,sin(xx.^2),':k')
```

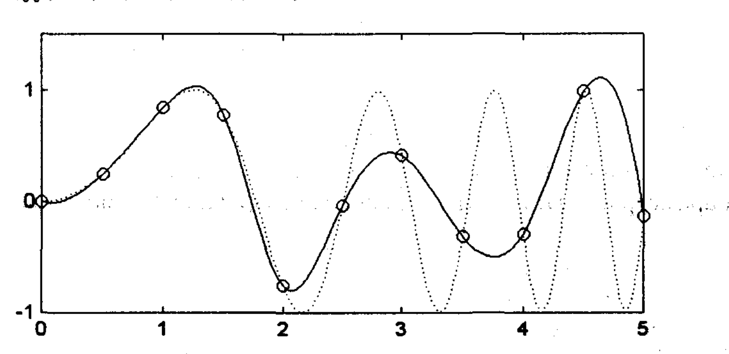


Рис. 15.6. Построение сплайна для незамкнутой кривой

На начальном участке сплайн-интерполяция неплохо описывает поведение аппроксимируемой кривой, а далее сказывается малое число использованных узлов.

Для примера разберем построение сплайн-интерполяции в случае замкнутой кривой $(x^2-1)^2+y^2=2$. Определим характерные точки кривой. Векторы p и q дают абсциссы и ординаты четырех точек верхней половины кривой:

```
» y0=sqrt(2); x0=sqrt(1+y0); p=[x0 1 0 -1]; q=[0 y0 1 y0];
```

Сформируем матрицу из двух строк (абсциссы и ординаты), добавив в начале и конце строк значения производных, что необходимо при построении сплайна для замкнутой кривой:

```
» pq=[0 p -p x0 0; pi/2 q -q 0 pi/2];
```

Вычислим сам сплайн, первый аргумент отвечает переменной параметризации:

```
» sp=spline(0:8,pq);
```

Теперь вычислим значения сплайна для интервала изменения переменной параметризации:

```
» tt=0:.1:8; hh=ppval(sp,tt);
```

Построим график, представив сплайн-интерполяцию сплошной линией и кружками — исходные точки:

```
» plot(hh(1,:),hh(2,:),p,q,'o',-p,-q,'o')
```

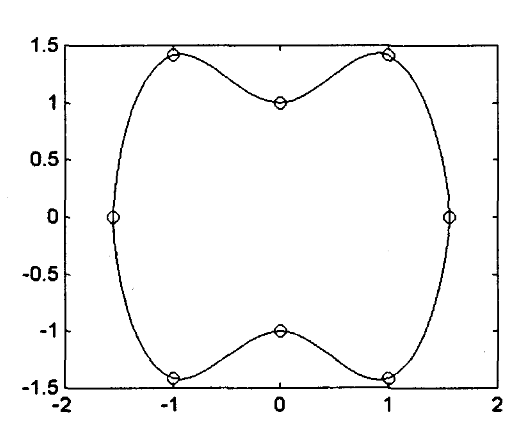


Рис. 15.7. Аппроксимация кривой сплайнами

Среди библиотек MATLAB для работы со сплайнами специально предназначен пакет The Spline Toolbox.

Для аппроксимации периодической функции используется команда `interpft(Z,N)`. Здесь параметр N задает число элементов возвращаемого массива. Приведем пример:

```
» T=3*pi; t=0:T; z=tanh(sin(t));
tp=0:.2:T; zp=interpft(z,length(tp));
tt=0:.1:T; zt=tanh(sin(tt));
plot(tt,zt,':k',t,z,'ok',tp,zp,'k')
```

На рис. 15.8 построен график исходной функции (пунктир) и отмечены точки, использованные для интерполирования (кружки), а также нанесена аппроксимирующая функцию кривая (сплошная линия).

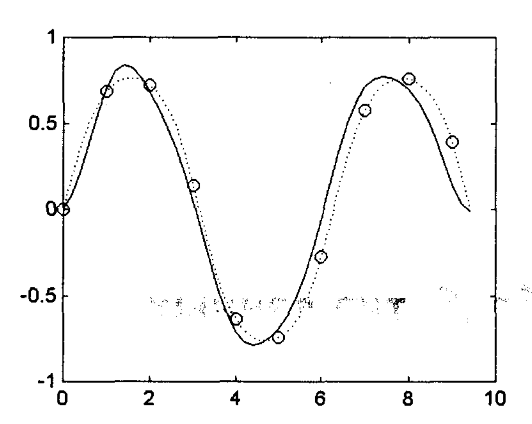


Рис. 15.8. Команда interpft и интерполяция периодической функции

Метод наименьших квадратов — очень эффективный способ аппроксимации данных. Для этой цели служит команда `polyfit(X,Z,N)`, вычисляющая полином степени `N`, для таблицы данных `X` и `Z`. Для получения аппроксимации используется матрица Вандермонда. Если потребуется вместо полиномов использовать другие базисные функции, то можно внести изменения в нужную функцию или написать собственный вариант метода наименьших квадратов.

Рассмотрим пример нахождения методом наименьших квадратов квадратичного полинома, наименее уклоняющегося от данных из таблицы. Подготовим векторы аргументов и значений функции:

```
» x=[0 .3 : 3]'; y=sin(x);
```

Вычислим коэффициенты аппроксимирующего кубического полинома при помощи стандартной функции:

```
» p=polyfit(x,y,3);
```

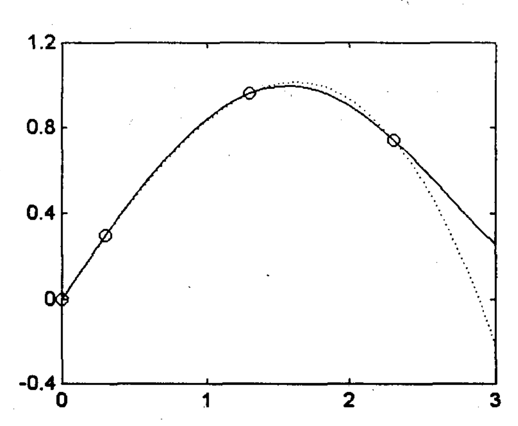


Рис. 15.9. Построение методом наименьших квадратов квадратичного полинома по таблице

Зададим матрицу для определения коэффициентов полинома с нечетными степенями 1, 3, 5 и вычислим решение:

```
» A=[x, x.^3, x.^5]; a=A\y;
```

Для изображения полученного полинома подготовим массивы:

```
» X=(0:.1:3)';Y=[X, X.^3, X.^5]*a; Yp=polyval(p,X);
```

Теперь нарисуем график построенного полинома сплошной кривой, пунктирной кривой дадим график синуса и выведем точки исходной таблицы кружками, см. рис. 15.9:

```
» plot(X,Y,'-',x,y,'o',X,Yp,':')
```

Анализ и обработка данных

Команды элементарного анализа данных (нахождение максимума и минимума, вычисление сумм, средних и медиан) были рассмотрены в главе 13 «Матричные вычисления». Продолжим изучение функций MATLAB и рассмотрим некоторые команды для статистической обработки, Фурье-анализа и фильтрации данных.

Функции корреляционного анализа приведены в табл. 15.7, в которой использованы следующие обозначения для входных параметров: X и Y – массивы данных.

Таблица 15.7. Команды статистической обработки данных

Команда	Назначение
cov(X,Y)	Вычисление матрицы ковариаций
corrcoef(X,Y)	Вычисление коэффициентов корреляции
std(X)	Стандартное квадратичное отклонение

Функция cov для вектора возвращает дисперсию, а для двумерного массива — матрицу ковариаций, при этом каждый столбец рассматривается как переменная, а строки — как наблюдения. Вектор стандартных отклонений может быть получен по команде sqrt(diag(cov(A))) или std. Обращение cov(X,Y) эквивалентно cov([X Y]), когда массивы X и Y имеют одинаковое число строк. Коэффициенты корреляции определяются функцией corrcoef.

Простой пример анализа девяти наблюдений двух периодических функций:

```
» x=pi/4*(1:9); X=[-7+sin(x);7+cos(x)]';
» C=cov(X), corrcoef(X), std(X)
C =
    0.5556    0.0556
    0.0556    0.5556
ans =
    1.0000    0.1000
    0.1000    1.0000
```

```
ans =
    0.7454    0.7454
```

Специальная библиотека команд собрана в пакет MATLAB Statistics Toolbox.

Преобразование Фурье и его дискретный вариант — важные компоненты многих расчетных процедур, включая анализ сигналов и обработку изображений. Эффективной реализацией дискретного преобразования является алгоритм быстрого преобразования Фурье FFT (Fast Fourier Transform). Набор функций MATLAB для работы с прямыми и обратными преобразованиями Фурье дан в табл. 15.8.

Таблица 15.8. Команды на основе Фурье-преобразования

Команда	Назначение
fft(X,N)	Одномерное дискретное преобразование Фурье
fft2	Двумерное дискретное преобразование Фурье
fftn	n-мерное дискретное преобразование Фурье
ifft	Обратное преобразование Фурье
ifft2	Обратное двумерное преобразование Фурье
ifftn	Обратное n-мерное преобразование Фурье
fftshift	Перегруппировка выходных массивов преобразований Фурье с размещением нулевой частоты в середине спектра

Быстрое преобразование Фурье позволяет анализировать временные ряды, определяя частоты периодических (регулярных) компонент сигнала и выделяя стохастический сигнал. Если длина входной последовательности является степенью двойки, то применяется алгоритм быстрого преобразования Фурье (БПФ), имеющий максимальную производительность. Специальная оптимизация для вещественных входных данных обеспечивает дополнительное ускорение обработки. Для данных иной длины используется алгоритм, требующий существенных затрат времени. Скорость работы можно повысить, указав дополнительным параметром длину обрабатываемой последовательности. Если длина входных данных больше указанного числа, то произойдет усечение данных, в противном случае недостающие элементы заполнятся нулями.

Рассмотрим примеры только для команды `fft` и вначале разберем случай с малым числом элементов. Зададим вектор и вычислим его преобразование Фурье:

```
» x=[2 4 3 5]; y=fft(x)
y =
  14.0000      -1.0000+1.0000i  -4.0000      -1.0000-1.0000i
```

Результат дается вектором с комплексными числами. Для четного числа (2n элементов) входных данных выходной массив всегда обладает следующей структурой: первый элемент дает сумму исходных данных, со второго по n-й элемент идут частоты, элемент n+1 (здесь третий) — это так называемая частота Найквиста, после которой следуют отрицательные частоты. Для вещественных входных данных это величины, сопряженные к элементам выходного массива со второго по n-й. При нечетном числе входных данных результатом будет сумма элементов и два набора комплексных чисел.

Рассмотрим пример анализа временного ряда, полученного суммированием двух периодических составляющих и случайной компоненты с нулевым средним:

```
» m=999;t=0:1/m:1;
x=sin(2*pi*90*t)+sin(2*pi*160*t)+randn(1, m+1);
```

Вычислим преобразование Фурье и определим мощность сигнала, приготовим также частотный диапазон для графика:

```
» y=fft(x); power=abs(y); freq=(0:m)*m/(m+1);
```

Теперь на рис. 15.10 приведем часть исходных данных (график слева) и результаты спектрального анализа (график справа). Для удобства ограничим отрезок частот при помощи команды axis:

```
» subplot(121), plot(x(1:100)),
subplot(122), plot(freq,power), axis([0 250 0 Inf])
```

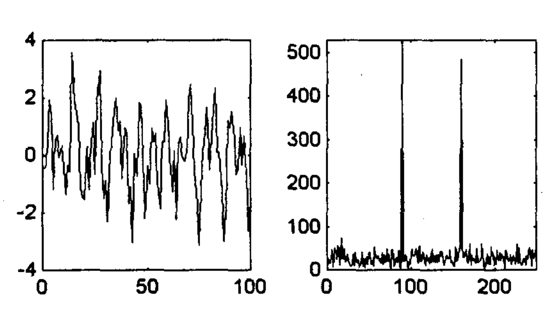


Рис. 15.10. Сигнал и его обработка при помощи команды fft

На полученном рисунке ясно видны две частоты периодических компонент сигнала, см. правую часть рисунка.

Для обработки сигналов в MATLAB имеется специальный пакет Signal Processing Toolbox.

Свертка двух сигналов эквивалентна умножению преобразований Фурье этих сигналов и осуществляется при помощи команды z=conv(x,y), для обратной операции имеется команда [p,q]=deconv(z,x). Для двумерных массивов свертка производится при помощи команды conv2.

Фильтрация входных данных помогает их усреднять, что бывает важно для временных рядов. Одномерная и двумерная фильтрация данных поддерживается соответственно командами filter и filter2. Основной вариант обращения к команде y=filter(b,a,x) означает реализацию следующего разностного уравнения:

```
a(1)y(n)=
b(1)x(n)+...+b(nb+1)x(n-nb)-a(2)y(n-1)-...-a(na+1)y(n-na)
```

Здесь na и nb есть число элементов соответственно массивов a и b, а выходной массив y представляет собой комбинацию исходного и получаемого массивов данных.

Приведем пример. Приготовим массив данных, вычислив значения синуса на грубой сетке, и назначим фильтр, усредняющий данные по пяти точкам:

```
» t=1:100;x=sin(2*t);a=1; b=ones(5,1)/5;
```

Теперь запустим команду и нарисуем исходные и отфильтрованные данные:

```
» z=filter(b,a,x); plot(t,x,t,z,'o')
```

Из-за большого шага кривая исходных данных (сплошная линия) не похожа на синусоиду, а фильтр срезает амплитуды, и получаются две синусоиды большого периода, см. рис. 15.11.

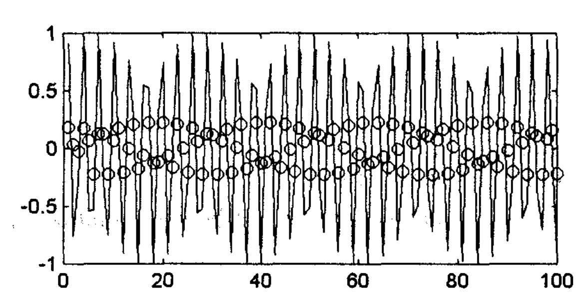


Рис. 15.11. Действия фильтра (команда filter)

Интегрирование дифференциальных уравнений

Для решения задачи с начальными данными (задача Коши) имеется ряд функций, позволяющих интегрировать системы неавтономных дифференциальных уравнений первого порядка y'=F(t,y) с независимой переменной t (обычно это время) и вектором неизвестных функций y. В стандартном наборе MATLAB для интегрирования нежестких систем могут быть использованы функции ode23 и ode45, а для жестких уравнений имеются функции ode113, ode15S и ode23S. Описание используемых методов численного интегрирования задачи Коши можно найти в книге [73].

Обращение ко всем функциям-интеграторам практически однотипно, поэтому для изложения будем использовать символическое имя ode_solver, а для примеров будем применять команду ode45 (метод Рунге–Кутта порядка 4/5). Итак, интегрирование уравнения или системы уравнений с правой частью, вычисляемой при помощи функции F, производится по команде

```
[T,Y]=ode_solver("F",TSPAN,Y0,OPTIONS,PARAMS)
```

Здесь TSPAN=[T0 TFINAL] (или диапазон значений) — массив, определяющий интервал интегрирования (значения переменной t, для которых будет вычислено решение), Y0 — вектор-столбец начальных условий, а дополнительные параметры OPTIONS и PARAMS, используемые соответственно для задания параметров и передачи параметров интегрируемой системы, могут отсутствовать.

Функция F(t,y) должна быть подготовлена заранее. Ее назначение — вычислять правую часть системы для вектора y при значении независимой переменной t и возвращать результат в виде вектора-столбца.

Результаты решения задачи Коши размещаются в матрице Y, причем каждая строка есть вектор решения, отвечающий значению независимой переменной из векто-

ра-столбца T. Для того чтобы получить решения в заданные моменты времени t0, t1, ..., tfinal, следует определить аргумент

```
TSPAN=[t0 t1 ... tfinal];
```

Через параметр OPTIONS могут быть определены значения абсолютной и относительной погрешности, максимальный шаг интегрирования и другие величины. Если аргумент OPTIONS отсутствует, то интегрирование ведется при значениях, заданных по умолчанию. Например, значение относительной погрешности RelTol равно 1e-3, и все компоненты вектора абсолютной погрешности (погрешность для каждой компоненты) AbsTol одинаковы — 1e-6. Для изменения этих и других параметров применяется функция odeset, а возможные параметры и ряд переключателей перечислены в следующих далее таблицах.

Заголовок функции правой части системы дифференциальных уравнений должен быть оформлен следующим образом:

```
function out=F(t,y,flag,PARAMS)
```

Здесь flag — строковая переменная для отработки событий. Заметим, что в перечне параметров функции ode45 (или другого интегратора) должен присутствовать параметр OPTIONS, хотя бы в виде пустого массива [], если никаких изменений параметров интегрирования не требуется.

Массивы TSPAN, Y0 и OPTIONS можно задавать в файле, вычисляющем правую часть F(t,y). Тогда TSPAN или Y0 при обращении к ode45 можно опустить; в этом случае интегратор перед началом работы обратится к функции F, чтобы определить недостающие параметры:

```
[TSPAN,Y0,OPTIONS]=F([],[],'init').
```

«Пустые» параметры можно опустить, и тогда обращение к функции-интегратору принимает кратчайшую форму:

```
ode_solver("F")
```

Для создания и последующего изменения параметров интегратора имеется функция odeset. Результатом выполнения команды будет структура OPTIONS:

```
OPTIONS=odeset("NAME1",VAL1,'NAME2',VAL2,...)
```

Здесь параметрам с зарезервированными именами NAME1, NAME2, ... присваиваются значения VAL1, VAL2, ... При этом не обязательно указывать все параметры, значения не упомянутых будут присвоены по умолчанию. Более того, достаточно указать несколько первых букв, идентифицирующих имя изменяемого параметра, а регистр при этом не важен.

Чтобы изменить параметры интегратора на основе существующей структуры OLD, следует ввести:

```
OPTIONS=odeset(OLD,'NAME1',VALUE1,...)
```

Для объединения назначений старой (OLD) и новой (NEW) структур параметров используется команда, по которой новые значения замещают значения параметров старой структуры:

```
OPTIONS = odeset(OLD,NEW)
```

Запуск функции odeset без параметров выводит все параметры с присвоенными им значениями.

Перечислим параметры, указав в скобках тип и значение, присвоенное по умолчанию.

Таблица 15.9. Параметры команд интегрирования дифференциальных уравнений

Имя	Назначение
RelTol	Величина относительной погрешности (скаляр, 1e-3)
AbsTol	Величина абсолютной погрешности (скаляр или вектор, 1e-6). Когда задана скалярная погрешность, то все компоненты вектора решения будут оцениваться одинаково, а при задании вектора — каждая компонента будет оцениваться по соответствующей компоненте вектора погрешности
Refine	Коэффициент уточнения (целое число, равное 4 для ode45 и 1 для остальных интеграторов) увеличивает число выводимых точек; не действует, если length(TSPAN)>2
OutputFcn	Имя функции, к которой интегратор обращается после успешного шага; при запуске без параметров это имя "odeplot"
OutputSel	Вектор индексов компонент вектора решения, передающихся в OutputFcn, по умолчанию передаются все компоненты
MaxStep	Верхняя граница шага интегрирования (положительное число, по умолчанию задается десятая часть интервала интегрирования)
InitialStep	Начальное значение шага интегрирования (положительное число)
MaxOrder	Максимальный порядок для функции ode15s (от 1 до 5)
Mass	Наличие матрицы масс (none | M | M(t) | M(t,y)); по умолчанию указано 'none', т. е. функция с правой частью системы ОДУ при обращении F(t,y,'mass') возвращает матрицу масс. Для 'M' матрица масс постоянна; для 'M(t)' матрица масс зависит от времени; для 'M(t,y)' матрица масс зависит от времени и координат

Иногда требуется прервать расчет при реализации некоторого события. В этом случае свойство Events должно быть объявлено ("on") и в файле правой части системы дифференциальных уравнений должна быть определена функция события, которая будет вычисляться при обращении из интегратора F(T,Y,'events'). В процессе расчета задачи Коши будут отслеживаться переходы через нуль для функции события и формироваться дополнительные векторы TE, YE, IE. В векторе-столбце TE будут храниться моменты времени, для которых произошли события, матрица YE составится из строк с решениями, соответствующими элементам вектора TE, а вектор IE будет содержать информацию о том, какое событие произошло.

Функция odeplot позволяет параллельно с расчетом наблюдать графики зависимости от времени всех (или указанных при помощи параметри "OutputSel") компонент вектора решения; при этом должна быть задействована возможность обращения к функции вывода:

```
OPTIONS=odeset("OutputFcn",'odeplot')
```

Аналогично функции odephas2 и odephas3 позволяют отображать в процессе расчета две или три компоненты вектора решения в виде соответствующего фазо-

вого портрета, а odeprint позволяет выводить числовую информацию о решении.

Имеется несколько переключателей, управляющих дополнительными возможностями интегратора. Допустимы значения on или off, в скобках указано значение, принятое по умолчанию.

Таблица 15.10. Параметры-переключатели

Имя	Назначение
Stats	Выведение статистики вычислительных затрат (off)
Jacobian	Наличие матрицы Якоби (off); если указано "on", то функция с правой частью системы ОДУ при обращении F(t,y,'jacobian') выводит матрицу Якоби dF/dy
JConstant	Матрица Якоби не зависит от решения (off)
JPattern	Матрица Якоби разреженная (off); если указано "on", то функция с правой частью системы ОДУ при обращении F(t,y,'jpattern') выводит матрицу Якоби dF/dy как разреженную, то есть только ненулевые элементы
Vectorized	Правая часть системы ОДУ векторизована (off); если указано "on", то при обращении F(t,[y1 y2 ...]) выводится [F(t,y1) F(t,y2) ...]
Events	Переключатель событий (off); если указано "on", то функция с правой частью системы ОДУ при обращении F(t,y,'events') возвращает значения функции; см. подробности в odefile
MassConstant	Матрица масс не зависит от времени (off)
BDF	Использование формулы дифференцирования назад в функции ode15s (off)
NormControl	Оценка ошибки по норме решения (off)

Для оценки погрешности по норме norm(e)<=max(RelTol*norm(y),AbsTol) нужно установить переключатель "on", по умолчанию используется более жесткий контроль точности.

Узнать текущее значение параметра NAME можно при помощи функции odeget, например:

```
VAL=odeget(OPTIONS,'NAME'),
```

при этом допускается указание первых букв имени, позволяющих идентифицировать параметр. Если в текущем сеансе никаких назначений не было, то возвращается значение, принятое по умолчанию.

В качестве примера рассмотрим систему Лоренца и подготовим в файле lor.m функцию вычисления правой части системы:

```
function f=lor(t,y,flag,sigma,r,b)
% lor - right hand side of Lorenz equation
% Правая часть системы Лоренца
f=[sigma*(y(2)-y(1)); ...
-y(2)+(r-y(3))*y(1); -b*y(3)+y(1)*y(2)];
```

Зададим параметры системы и определим значения для относительной и абсолютной погрешности, используя команду odeset:

```
» s=10.; r=24.5; b=8./3.;
opt=odeset("RelTol",1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
```

Вычислим решение, стартуя из точки `[0 1 1]`, и нарисуем график зависимости от времени второй и третьей переменных, см. рис. 15.12:

```
» [T,Y]=ode45("lor",[0 20],[5 7 9],opt,s,r,b);
plot(T,Y(:,2:3))
```

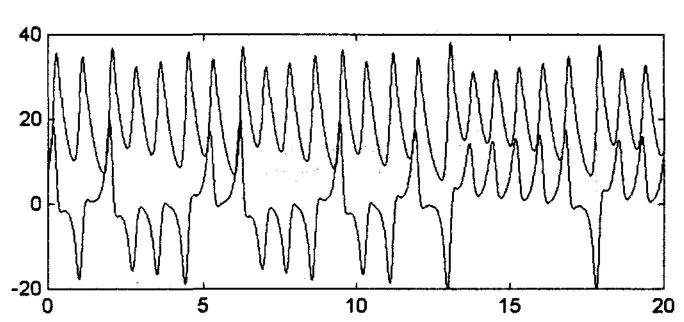


Рис. 15.12. Интегрирование системы Лоренца

Выберем последнюю полученную точку в качестве начальной и продолжим расчет:

```
» y0=Y(end,:), [T,Y]=ode45("lor",[0 30],y0,opt,s,r,b);
```

Нарисуем траекторию в фазовом пространстве системы (трехмерное пространство для системы Лоренца), см. рис. 15.13.

```
» plot3(Y(:,1),Y(:,2),Y(:,3));
axis([-10 10 -10 10 0 50]), grid off
```

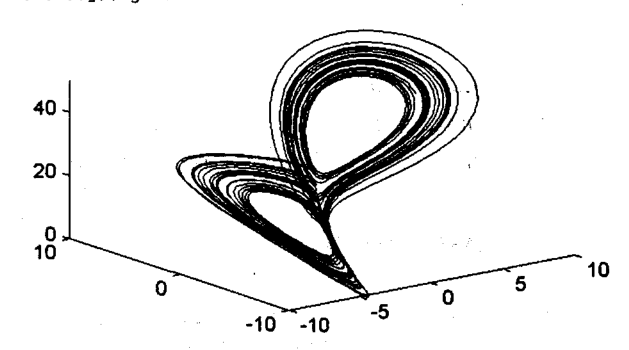


Рис. 15.13. Траектория в фазовом пространстве

Также имеются команды оживления фазовых картин: двумерной — `comet` и трехмерной — `comet3`. В качестве иллюстрации вычислим траекторию для системы Лоренца:

```
» [T,Y]=ode45("lor",[0:.02:20],[-5 -4 19],[],10,29,8/3);
```

и нарисуем фазовый портрет для первой и третьей переменных, см. рис. 15.14:

```
» comet(Y(:,1),Y(:,3))
```

Ту же фазовую траекторию можно нарисовать при помощи команды

```
» plot(Y(:,1),Y(:,3),'k',Y(end,1),Y(end,3),'ok')
```

Различные примеры приготовленных правых частей содержатся в файлах orbitode (ограниченная проблема трех тел), orbt2ode (задача двух тел), rigidode (уравнения Эйлера), vdpode (уравнение Ван дер Поля).

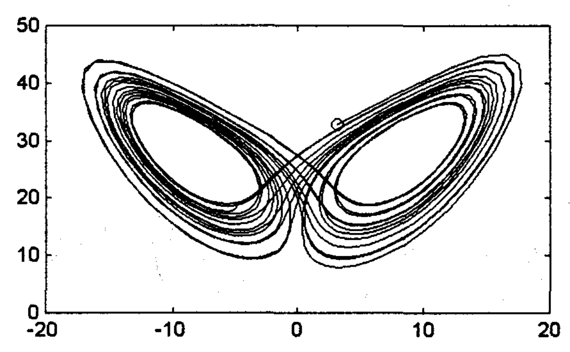


Рис. 15.14. Фазовая траектория, выведенная командой comet

В заключение приведем еще одну форму обращения к команде ode_solver решения дифференциального уравнения:

```
[T,Y,S]=ode_solver("F",TSPAN,Y0,OPTIONS,PARAMS)
```

По сравнению с ранее рассмотренными обращениями здесь в левой части появился параметр S, в котором накапливается статистика о ходе решения задачи. Это массив из шести элементов, в которых содержится следующая информация:

- число успешных шагов;
- число неудачных попыток;
- число вычислений правой части;
- число вычислений матрицы Якоби;
- число проведенных факторизаций;
- число решений систем линейных алгебраических уравнений.

Решение краевых задач

В MATLAB 6 добавлена команда bvp4c для решения одномерных краевых задач, задаваемых системой обыкновенных дифференциальных уравнений первого порядка. Решение краевой задачи на отрезке ищется при помощи метода коллокаций, для дискретизации используются формулы Лобатто и конечные разности. Как обычно в MATLAB, действуют разумные назначения точности, допустимого числа разбиений и ряда других параметров. Для получения их значений имеется команда bvpget, а изменить их позволяет функция bvpset. Кроме того, имеются вспомогательные функции для формирования начального приближения (bvpinit) и получения решения в заданных точках (bvpval).

Для использования функции bvp4c краевая задача должна быть представлена в виде системы обыкновенных дифференциальных уравнений первого порядка относительно неизвестной вектор-функции y(x) переменной x и параметров p:

```
y'=f(x,y,p)
```

Задача рассматривается на отрезке [a,b], и краевые условия формулируются следующим образом:

```
c(y(a),y(b),p)=0
```

Здесь c — заданная функция от значений функций в точках a и b. Обращение к bvp4c имеет следующий вид:

```
SOL=bvp4c(ODE,BC,INIT,OPT,P1,P2,...)
```

Здесь ODE и BC обозначают имена функций, предназначенных для вычисления правой части системы и краевых условий соответственно. Структура INIT с зарезервированными полями x и y дает начальное приближение к решению, узлы идут в порядке возрастания, так что INIT.x(1)=a и INIT.x(end)=b. Структура INIT может быть сформирована при помощи команды bvpinit. Ниже в примере указаны два способа формирования структуры, задающей начальное приближение.

Имя OPT обозначает переменную с параметрами расчета, а параметры задачи P1, P2, ... должны передаваться в функцию вычисления правой части и функцию, задающую краевые условия. Способ передачи аналогичен рассмотренному ранее для команд решения задачи с начальными данными. Если в системе имеются параметры P1,P2,..., то место параметров OPT при обращении к bvp4c должно быть занято хотя бы пустым массивом []. Если интегрируется система без параметров и нет необходимости менять параметры расчета, то обращение к функции bvp4c может выглядеть следующим образом:

```
SOL=bvp4c(ODE,BC,INIT)
```

Рассмотрим оформление заголовков функций вычисления правой части ODE и задания краевых условий BC. Входными параметрами для функции вычисления правой части с именем ODE являются точка отрезка X, вектор значений в этой точке Y и параметры задачи P1, P2, ... Выходные данные представлены вектором DY:

```
DY=ODE(X,Y,P1,P2,...)
```

Краевые условия обрабатываются функцией

```
C=BC(Ya,Yb,P1,P2,...)
```

Здесь векторы Ya и Yb дают значения вектора на разных концах интервала, а выходным параметром будет вектор C.

Решение SOL вычисляется в виде структуры, поля которой фиксированы и имеют следующий смысл: x — координаты узлов разностной сетки, y — аппроксимация решения в узлах, yp — аппроксимация первой производной решения в узлах, и parameters — значения неизвестных параметров. Последний параметр позволяет решать задачи по определению собственных чисел и функций (см. файл **mat4bvp.m** из каталога демонстраций **\toolbox\matlab\demos**, где bvp4c применяется для вычисления четвертой собственной функции уравнения Матье).

Для изменения характеристик расчета можно задать соответствующие параметры OPT: AbsTol дает величину абсолютной, а RelTol — относительной погрешности, параметр FJacobian указывает на наличие аналитического выражения для якобиана системы, а BCJacobian — якобиана для системы краевых условий, Nmax задает предельный порядок получающейся системы линейных алгебраических уравнений,

Stats определяет параметр решения. Порядок системы есть число неизвестных функций, умноженное на число узлов, по умолчанию число Nmax равно 1000. Команда bvpset() распечатывает имена всех параметров и их возможные значения, отмечая используемые по умолчанию фигурными скобками. В процессе расчета происходит измельчение сетки для достижения заданной точности.

Рассмотрим пример вычисления решений следующей краевой задачи:

y»+y²=0, y(0)=0, y(2)=-1

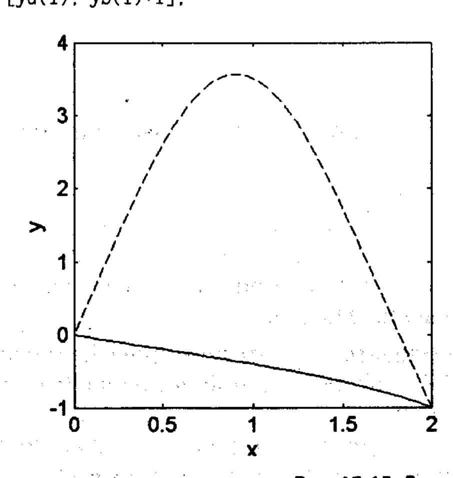
Оформим пример с использованием подфункций subfunction. В одном файле поместим операторы подготовки и вычисления решения, а также вспомогательные функции вычисления правой части ode, задания краевых условий bc и подготовки начального приближения fun. При передаче в качестве параметров имен подфункций снабжаем их префиксом @. Так, при первом вызове функции bvpinit используем функцию fun для задания начального приближения решения в виде синусоиды и производной решения в виде косинусоиды. Для построения второго решения задаем начальное приближение константами для каждой компоненты решения. Затем дважды вызываем функцию решения краевой задачи bvp4c для нахождения решений и используем команду bvpval для получения массивов решений в заданных точках.

```
function bvp_demo  %Two solution
b=2; n=8; x=b/n*[0:n];
Yinit=bvpinit(x,@fun);  Y1=bvp4c(@ode,@bc,Yinit);
Yinit=bvpinit(x,[-1 0]); Y2=bvp4c(@ode,@bc,Yinit);
x=b/50*[0:50]; y1=bvpval(Y1,x); y2=bvpval(Y2,x);
subplot(121), plot(x,y1(1,:),x,y2(1,:));
subplot(122), plot(x,y1(2,:),x,y2(2,:));

function Y=fun(x)
Y=4*[sin(2.*x); cos(3.*x)];

function dydx=ode(x,y)
dydx=[y(2); -y(1).^2];

function f=bc(ya,yb)
f=[ya(1); yb(1)+1];
```



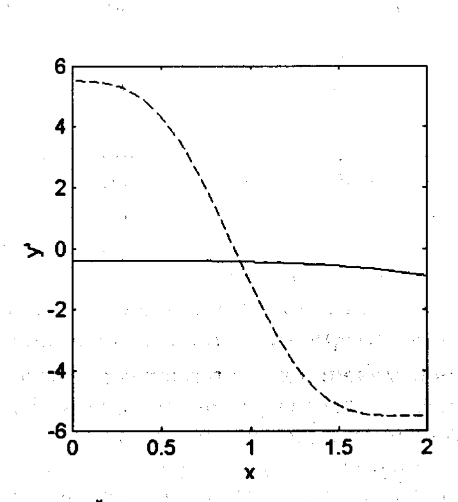


Рис. 15.15. Два решения краевой задачи

Результат выполнения функции bvp_demo представлен на рис. 15.15, в левой части даны сами решения — функции y(x), а на правом рисунке приведены графики их производных. Отметим, что для вычисления первого решения (пунктир) потребовалось 19 узлов сетки, включая концы интервала, а для вычисления второго решения (сплошная кривая) хватило 10 узлов.

Решение начально-краевых задач параболического типа

В MATLAB 6 появилась также команда pdepe для решения системы уравнений в частных производных параболического типа, с одной пространственной координатой. Общий вид системы относительно неизвестной вектор-функции $U(x,t)$ может быть представлен следующим уравнением:

$$C\frac{\partial U}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m F\right) + S$$

Здесь матрица C, векторы F и S зависят от переменных $x, t, U, \frac{\partial U}{\partial x}$. Матрица C является диагональной, по крайней мере, один элемент ее должен быть ненулевым. Для уравнения теплопроводности вектор F определяет поток, а вектор S описывает источники. Введение параметра m позволяет решать одномерные задачи для радиальной координаты в цилиндрической или полярной (m=1), а также сферической (m=2) системах координат.

Начальные условия определяются для вектора неизвестных:

$$U(x, t_0) = U_0(x)$$

Краевые условия задаются в следующем виде:

$$P(x,t,U) + Q(x,t)F(x,t,U,\frac{\partial U}{\partial x}) = 0$$

Обращение к функции решения системы уравнений в частных производных имеет вид:

```
SOL=pdepe(m,PDE,IC,BC,XX,TT,OPT,P1,P2,...)
```

Здесь параметр m определяет систему координат: 0 отвечает декартовым координатам, 1 — цилиндрическим, 2 — сферическим. Вычисление правой части рассматриваемой системы производится в функции PDE, данные о начальных условиях берутся из функции IC, а краевые условия задаются функцией BC.

Сетка по координате x на интервале [a,b] должна содержать не менее трех узлов и задается массивом XX, содержащим узлы в порядке возрастания координаты. Лучше использовать более мелкий шаг между узлами в тех областях, где решение меняется значительно, и более крупный шаг в местах плавного изменения решения. Для m>0 (цилиндрические или сферические координаты) не обязательно сгущать сетку в окрестности особой точки x=0, и при m>0 значение a не может быть отрицательным.

Для интегрирования задачи по времени в функции pdepe используются команды для решения задачи Коши (ode15s и др.), а применяемый метод и шаг интегрирова-

ния выбираются функцией pdepe. Массивом TT определяется набор временных слоев, в которых будут запомнены решения, таких слоев должно быть не менее трех. Некоторые расчетные параметры можно изменить при помощи команды odeset, см. описание в разделе, посвященном интегрированию задачи Коши. В частности, доступны такие параметры, как относительная (RelTol) и абсолютная (AbsTol) величины погрешности, начальный (InitialStep) и максимальный (MaxStep) шаги интегрирования.

Решение выводится в трехмерный массив SOL, причем первый индекс (i) отвечает временному слою t_i, второй индекс (j) определяет номер узла x_j, а третий индекс дает номер компоненты вектора решения. Таким образом, элемент SOL(i,j,k) дает значение решения $U^k(x^j, t^i)$.

Рассмотрим заголовки функций вычисления правой части (PDE) и задания начальных (IC) и краевых (BC) условий. Входными параметрами для функции вычисления правой части с именем PDE являются точка отрезка X, вектор значений в этой точке Y и параметры задачи P1, P2, ... Выходные данные представлены диагональной матрицей C и векторами F, S:

```
[C,F,S]=PDE(X,T,U,UXP1,P2,...)
```

Начальные условия даются функцией:

```
U=IC(X,P1,P2,...)
```

Краевые условия обрабатываются функцией:

```
[Pa,Qa,Pb,Qb]=BC(a,Ua,b,Ub,T,P1,P2,...)
```

Здесь векторы Ua и Ub дают значения вектора на разных концах интервала [a,b], T — время, а выходные параметры представлены векторами на левом (Pa, Qa) и правом (Pb, Qb) концах интервала.

В дополнение к pdepe имеется команда pdeval для пересчета решения на другую сетку и получения массива решения пространственных производных. Обращение к этой команде имеет следующий вид:

```
[Uout,DUout]=pdeval(m,XX,U,Xout)
```

Здесь U — решение, полученное на сетке XX, Uout и DUout есть соответственно функция и ее пространственная производная на новой сетке Xout.

Рассмотрим применение команды pdepe для решения уравнения теплопроводности с источником (p — параметр):

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+pe^{-t}$$

Начальное распределение температуры есть $u(x,0) = p+\sin(x)$ и заданы следующие краевые условия: Дирихле (первого рода) на левом конце и Неймана (второго рода) на правом конце.

Создадим m-функцию pde_demo, в которой будут использованы операторы подготовки и вычисления решения, а также вспомогательные функции вычисления правой части pde, задания начальных (ic) и краевых (bc) условий. Указывая имена этих подфункций в качестве параметров команды pdepe, используем префикс @. Обозначения переменных практически повторяют использованные для описания

параметров команды. Концы отрезка изменения переменной x есть a и b, p — параметр, число ячеек на отрезке — n, переменная xx определяет узлы сетки по x, а переменная tt задает временные слои:

```
function pde_demo
m=0; a=0; b=3*pi/2; p=1;
n=16; xx=linspace(a,b,n); tt=0:.2:3;
sol=pdepe(m,@pde,@ic,@bc,xx,tt,[],p); u = sol(:,:,1);

subplot(121), mesh(xx,tt,u), )
axis([a b tt(1) tt(end) 0 1.5]), xlabel("x"), ylabel("t")
subplot(122),
plot(xx,u(end,:),'ok',xx,exp(-tt(end))*(p+sin(xx))),
xlabel("x"), ylabel(strcat("u(x,",num2str(tt(end)),')')),
axis tight

function [C,F,S] = pde(x,t,u,DuDx,p)
C=1; F=DuDx; S=-p*exp(-t);

function u0 = ic(x,p)
u0=p+sin(x);

function [Pa,Qa,Pb,Qb] = bc(a,ua,b,ub,t,p)
Pa=ua-p*exp(-t); Qa=0; Pb=0; Qb=1;
```

На рис. 15.16 приведены графики, полученные в результате запуска функции pde_demo и последующей доработки изображения средствами меню графического окна. А именно были передвинуты метки осей, убрано цветовое оформление вывода команды mesh и трехмерная картина тепловой диффузии (слева) повернута для получения лучшего ракурса. График справа дает распределение температуры на последнем из рассчитанных временных слоев, сплошной линией представлено имеющееся для данной задачи точное решение, а кружками — результаты расчета при помощи функции pdepe. Видно, что при остывании стержня сохраняется форма начального распределения.

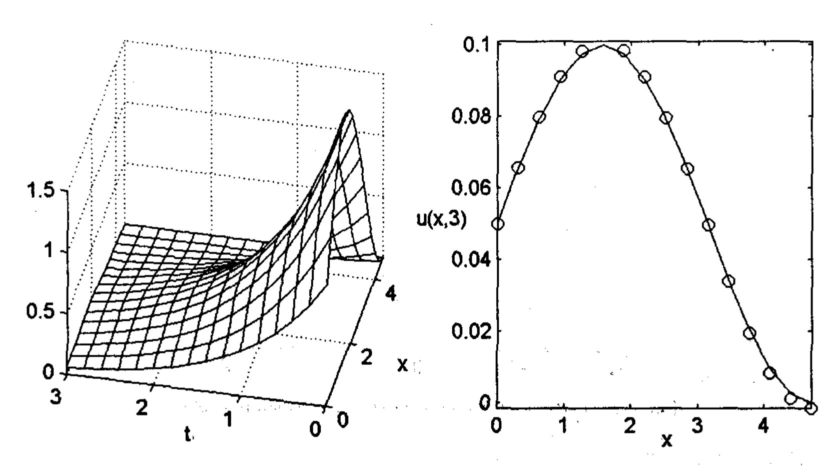


Рис. 15.16. Решение уравнения теплопроводности

Перед использованием команды pdepe полезно также ознакомиться с имеющимися в поставке MATLAB демонстрационными примерами.

Функции геометрического анализа

Примером того, насколько разнообразны возможности даже стандартного набора пакета MATLAB, служит набор геометрических функций для работы с многоугольниками и триангуляции плоской области.

Таблица 15.11. Команды работы с многоугольниками и триангуляции

Имя	Назначение
polyarea	Вычисление площади многоугольника
inpolygon	Определение попадания точки в многоугольник
convhull	Вычисление выпуклой оболочки
delaunay	Триангуляция Делоне
voronoi	Построение диаграммы Вороного

Приведем пример, иллюстрирующий работу этих команд. Многоугольник в виде звезды зададим набором вершин с абсциссами xp и ординатами yp:

```
» xp=[];yp=[];ang=2*pi/5;
                for k=0:4,
                xp=[xp,2*sin(ang*k),.9*sin(ang*k+.5*ang)];
                yp=[yp,2*cos(ang*k),.9*cos(ang*k+.5*ang)];
                end;
                xp=[xp,xp(1)]; yp=[yp,yp(1)];
```

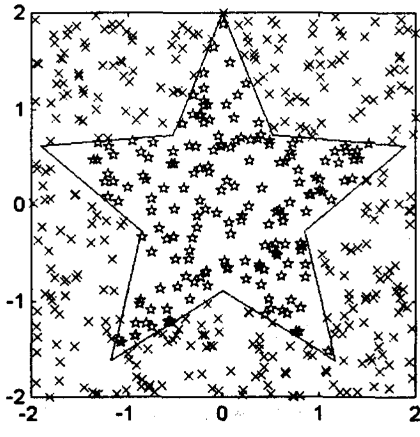


Рис. 15.17. Многоугольник и множество случайных точек

Вычислим площадь этого многоугольника:

```
» polyarea(xp,yp)
```

```
ans =
    5.2901
```

Теперь зададим 500 точек случайным образом и определим их принадлежность многоугольнику. Затем нарисуем эти точки, помечая попавшие внутрь многоугольника звездочками, а остальные — крестиками, см. рис. 15.17:

```
» x=4*rand(500,1)-2; y=4*rand(500,1)-2;
in=inpolygon(x,y,xp,yp);
plot(xp,yp,x(in),y(in),'pr',x(~in),y(~in),'xb')
```

Для триангуляции области используется функция delaunay, реализущая алгоритм Делоне построения сетки из треугольников. Для этого необходимы также функции griddata — для интерполяции данных и voronoi — для построения диаграммы Вороного. Функциия dsearch используется для нахождения ближайшей соседней точки, a tsearch — для отыскания треугольников, содержащих указанные точки.

Возможны различные варианты обращения к функции delaunay, как практически к любой из функций MATLAB. Рассмотрим простейший случай: входные параметры есть точки определенного ранее многоугольника в форме звезды. Результатом триангуляции является массив, каждая строка которого содержит индексы трех вершин треугольника так, чтобы ни одна из исходных точек не содержалась внутри него:

```
» Tri=delaunay(xp,yp); size(Tri)
ans =
    13     3
```

Теперь построим полученную систему треугольников, см. рис. 15.18:

```
» hold on,
for k=1:size(Tri,1), tt=Tri(k,:); plot(xp(tt),yp(tt)), end
```

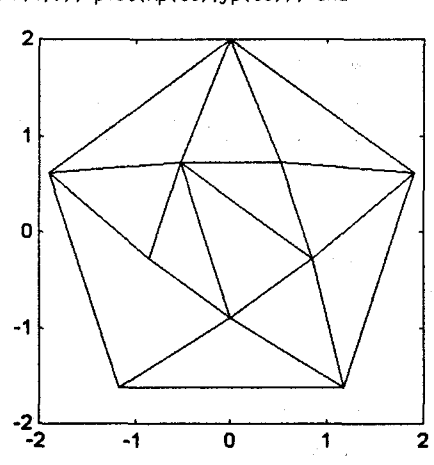


Рис. 15.18. Триангуляция области

В заключение продемонстрируем построение диаграммы Вороного и оболочки, охватывающей набор точек, задающих звезду. Для этого используем графическую функцию subplot, которая позволяет построить в одном окне несколько графиков и описана в главе 14 «Графика в MATLAB», см. рис. 15.19:

```
» subplot(1,2,1),
[vx,vy]=voronoi(xp,yp); plot(vx,vy,'-k',xp,yp,'ok')
subplot(1,2,2), axis square
k=convhull(xp,yp), plot(xp,yp,'o',xp(k),yp(k),'-')
```

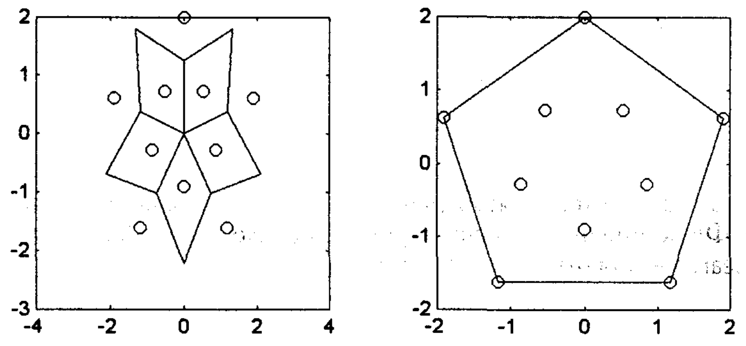


Рис. 15.19. Диаграмма Вороного (слева) и результат выполнения команды convhull (справа)

Специальные математические функции

Помимо множества функций численного анализа возможности пакета характеризуются наличием большого числа качественно реализованных специальных математических функций. В табл. 15.12 представлен перечень наиболее важных из них. Полную информацию о специальных математических функциях можно получить, вызвав справку

```
help specfun
```

Таблица 15.12. Специальные математические функции

Имя	Описание	Имя	Описание
airy	Функция Эйри	erf	Функция ошибки
bessel	Функция Бесселя	erfc	Дополнительная функция ошибки
besseli	Модифицированная функция Бесселя первого рода	erfcx	Масштабированная дополнительная функция ошибки
besselj	Функция Бесселя первого рода	erfinv	Обратная функция ошибки
besselk	Модифицированная функция Бесселя второго рода	expint	Интегральная показательная функция
bessely	Функция Бесселя второго рода	gamma	Гамма-функция
beta	Бета-функция	gammainc	Гамма-функция неполная
betainc	Бета-функция неполная	gammaln	Логарифм от гамма-функции
betaln	Логарифм от бета-функции	legendre	Присоединенная функция Лежандра

продолжение

Таблица 15.12 (*продолжение*)

Имя	Описание	Имя	Описание
ellipj	Эллиптическая функция Якоби	log2	Логарифм по основанию 2
ellipke	Полный эллиптический интеграл	pow2	Степенная функция по основанию 2

Команда log2 выводит представление числа в форме мантисса (число от 0,5 до 1) и порядок, а pow2 есть обратная ей команда:

```
» [m,p]=log2(4)
m =
  5.0000e-001
p =
     3
» pow2(m,p)
ans =
     4
```

Ряд математических функций становится доступным при подключении символьного анализатора Maple (Symbolic Math Toolbox), работа с которым описана в главе 17 «Расширения MATLAB». Для получения справки по поводу интересующей функции name следует выполнить команду help sym/name, например для справки о гамма-функции нужно набрать в командной строке help sym/gamma.



Программирование в MATLAB

Начальные сведения о программировании на языке MATLAB изложены в главе 12 «Элементы языка MATLAB», где описаны синтаксис и типы данных, основные операторы и программирование функций.

Эта глава посвящена вопросам программирования в системе MATLAB. Здесь описаны команды ввода-вывода, объектно-ориентированное программирование, отладка функций и профилирование разработанных программ, рассмотрены возможности компилятора MATLAB и разработка mex-файлов, обсуждено программирование интерфейса.

Последние версии пакета существенно расширили возможности MATLAB как интегрированной среды программирования. При помощи собственного компилятора MATLAB или стандартных компиляторов, установленных на компьютере, можно использовать существующее программное обеспечение на языках С и Фортране без его переписывания, а также ускорять вычисления за счет большей скорости выполнения циклов и других операций у откомпилированных модулей. В MATLAB поддерживается интерфейс API (Application Program Interface), что позволяет вызывать из MATLAB программы, написанные на С и Фортране, обмениваться данными с другими приложениями, устанавливать связи клиент-сервер между MATLAB и другими программами. Вызов MATLAB из Maple описан в главе 9 «Maple и другие программы».

Подробное описание всех возможностей, предоставляемых MATLAB для программирования, невозможно в рамках данной книги, поэтому в этой главе мы ограничимся их коротким изложением и демонстрационными примерами. Детальное описание программирования в среде MATLAB дано в документации [27], а на русском языке много полезной информации содержится в книге [17].

Команды ввода-вывода

В MATLAB поддерживается работа с внешними данными, подготовленными в различных форматах. Такая работа осуществляется при помощи набора эффектив-

ных команд. Для записи и считывания информации разработан специальный формат mat-файла, избавляющий пользователя от необходимости вникать в детали хранения данных. Более того, при помощи функций из библиотеки С можно создавать приложения с доступом к mat-файлам. Также имеется набор команд для форматного ввода и вывода информации, для записи бинарных файлов и для работы с некоторыми типами стандартных файлов.

Команды load и save

Мощными и удобными в работе являются команда save для сохранения информации и команда load для ее считывания. По умолчанию данные записываются в стандарте mat-файла. Схематически обращение к команде save выглядит следующим образом:

```
save [ИМЯ] [-ПАРАМЕТРЫ] [ПЕРЕМЕННЫЕ]
```

В квадратных скобках даны необязательные параметры, при отсутствии которых запись производится согласно системным назначениям. Когда имя файла ИМЯ не указано, то данные будут сохранены в двоичном файле matlab.mat. Если расширение файла пропущено, то данные запишутся в файл ИМЯ.mat. Список величин, подлежащих записи, задается параметром ПЕРЕМЕННЫЕ, при отсутствии которого сохраняются все данные из рабочей области.

Дополнительные ПАРАМЕТРЫ могут принимать следующие значения:

- ascii — сохранение в виде ASCII-файла с мантиссой из восьми цифр;
- ascii-double — сохранение в виде ASCII-файла с шестнадцатизначной мантиссой;
- append — дозапись в существующий файл.

При сохранении информации в двоичном коде для записи вещественных чисел используется 8 байт, а для целочисленных переменных могут применяться следующие форматы: int8 (однобайтное целое число от −128 до 127), int16 (двухбайтное целое число) и int32 (четырехбайтное целое число).

Приведем примеры. Запись переменных d и e в файл abc.mat производится одной командой

```
» save abc d e
```

Запись переменных d и e в текстовый файл abc.txt

```
» save abc.txt -ascii d e
```

Назначения параметров для команды load аналогичны описанным ранее:

```
load [ИМЯ] [-ПАРАМЕТРЫ] [ПЕРЕМЕННЫЕ]
```

Из файла ИМЯ считываются только переменные, перечисленные в списке ПЕРЕМЕННЫЕ, а отсутствие списка вызовет считывание всех величин, сохраненных в файле ИМЯ.

Данные из ASCII-файла с именем NAME.ext можно считать по команде

```
» load NAME.ext
```

Сами данные должны быть организованы в виде массива с одинаковым числом элементов во всех строках, иначе при считывании данных будет выведено сообщение об ошибке. По прочтении данные будут помещены в массив с именем NAME.

Обращаться к командам load и save можно так же, как к функциям. Например, для сохранения в файле NAME.mat переменных d и e следует ввести

```
» save("NAME.mat",'d','e')
```

а для считывания соответственно

```
» load("NAME.mat",'d','e')
```

Использование вызова функции удобнее, если аргумент-строка формируется из нескольких частей. Например, можно предложить следующий фрагмент для считывания и последующей обработки данных дневных наблюдений за июнь, записанных в файлах June1.dat, June2.dat и т. д.:

```
» for d=1:30, dd=["June" int2str(d) ".dat"], load(dd)
% Обработка данных
end
```

То же самое, но с потерей эффективности (работает интерпретатор, а выше использовался вызов функции), можно реализовать при помощи следующего фрагмента:

```
» for n=1:30, s=["load June" int2str(n) ".dat"],eval(s),
% Обработка данных
end
```

Записать и считать переменные с общей частью в имени можно при помощи знака «*», например, чтобы сохранить все переменные, начинающиеся с сочетания abc, достаточно команды

```
» save abc*
```

Форматные операции ввода-вывода

Для операций ввода-вывода с использованием форматов данных и работы с бинарными файлами применяются команды, напоминающие стандартные функции языка C, см. табл. 16.1.

Перед началом записи или чтения нужный файл следует открыть при помощи команды

```
FI=fopen('NAME', 'FLAG')
```

Имя файла NAME должно содержать путь, если файл берется из каталога, не указанного в списке каталогов Path Browser. Строковая переменная 'FLAG' определяет тип файла и способ работы с ним: 't' — текстовый файл, 'b' — бинарный файл, 'r' — чтение, 'w' — запись, 'r+' — чтение и запись одновременно, 'a' — добавление в конец файла. Например, для чтения бинарного файла следует задать флаг 'rb', для записи текстового — 'wt'. Полученный в результате выполнения операции fopen числовой идентификатор FI используется в качестве параметра для функций fread и fwrite. Если открыть файл не удалось, то FI=-1. Чтобы закрыть файл, надо выполнить команду

```
fclose(FI)
```

Таблица 16.1. Список команд ввода-вывода

Имя	Назначение
fopen	Открытие файла
fclose	Закрытие файла
fread	Считывание данных из файла
fwrite	Запись данных в файл
fscanf	Считывание форматированных данных из файла
fprintf	Запись данных в файл или вывод на экран
fgetl	Считывание строки из файла без служебных символов
fgets	Считывание строки из файла со служебными символами
dlmread	Чтение данных с разделителями из текстового файла
textread	Чтение форматированных данных из текстового файла

Команда fread применяется для чтения бинарных файлов:

```
[A.COUNT]=fread(FI.SIZE.'PREC'.'SKIP')
```

Здесь A — имя матрицы, куда заносятся считываемые данные, а COUNT — число прочитанных элементов. Если дополнительный параметр SIZE отсутствует, то считывается весь файл, целое число N в качестве SIZE задает считывание N элементов и формирование вектора-строки, а при указании в качестве SIZE вектора [M.N] формируется матрица размера [M.N], причем параметр N может принимать значение inf. Параметр 'PREC' задает число бит для записи чисел. Например, текстовая строка 'float64' указывает, что данные следует записывать в формате двойной точности, по 8 байт на число, а четырехбайтовый формат записи целых чисел задается строкой 'integer*4'. Перечень допустимых форматов можно получить по справке help fread. Дополнительный параметр 'SKIP' определяет число байтов, которое должно быть пропущено после считывания очередного элемента.

Команда fwrite используется для записи бинарных файлов:

```
COUNT=fwrite(FI.PARA.'PREC'.'SKIP')
```

Здесь FI — идентификатор файла, PARA — имя записываемой переменной, а назначение остальных параметров то же, что и для команды fread.

Приведем пример. Откроем файл aaa.1 на запись бинарных данных и сохраним магическую матрицу размера 2×2:

```
» fid=fopen("aaa.1".'wb'). fwrite(fid.magic(2).'integer*4')
fid =
     3
ans =
     4
```

Допишем вещественное число и закроем файл:

```
» fwrite(fid.pi.'float64'): fclose(fid)
ans =
     0
```

Теперь считаем записанные данные:

```
» f=fopen("aaa.1",'r+');[aa,n]=fread(f,[2,2],'integer*4'),
aa =
     1     3
     4     2
n =
     4
» bb=fread(f,'float64'), cc=fread(f), fclose(f);
bb =
  3.1416e+000
cc =
     []
```

Действительно, в файле aaa.1 была записана матрица второго порядка и число π. При попытке считывания еще одного числа получен пустой массив. Если же при чтении не определить структуру матрицы или указать иной формат для чтения, то результатом будет вектор-строка из элементов, число которых и сами значения определяются форматом считывания. Например, для формата float64 получаем строку из трех элементов:

```
» f=fopen("aaa.1");aa=fread(f,'float64'); fclose(f);aa'
ans =
  8.4880e-314  4.2440e-314  3.1416e+000
```

При считывании с установками по умолчанию получается вектор из 24 целых чисел:

```
» f=fopen("aaa.1");aa=fread(f); fclose(f);aa'
ans =
  Columns 1 through 12
   1   0   0   0   4   0   0   0   3   0   0   0
  Columns 13 through 24
   2   0   0   0  24  45  68  84 251  33   9  64
```

Для считывания части данных используется функция fseek(FI,NUM,'FLAG'), служащая для перемещения на целое число NUM позиций, причем положительное NUM означает продвижение к концу файла, а отрицательное – к началу файла. После открытия файла указатель установлен на первый байт. После считывания очередного числа указатель переместится на столько байтов, каков формат чтения. Текущее положение выводится по команде ftell(FI). Для проверки достижения конца файла служит команда feof(FI). Для возврата в начало файла можно использовать функцию frewind(FI).

Бинарные файлы являются просто потоками байтов, а для работы с текстом, разбитым на строки, используются текстовые файлы. Такие файлы могут содержать как текст, так и числовую информацию, которую для записи предварительно следует перевести в строковые переменные при помощи команд num2str и int2str. Для чтения информации из текстовых файлов применяются функции fgetl, fgets и fscanf. Для записи служит функция fprintf, причем если файловый идентификатор равен единице или отсутствует, то вывод идет на экран. Для чтения и записи применяются форматные спецификации, см. табл. 16.2.

Приведем пример. Подготовим вектор-строку

```
» y=1:4
y =
     1     2     3     4
```

и организуем вывод на экран

```
» fprintf("%7.2f %12.7f\n",y);
   1.00    2.0000000
   3.00    4.0000000
```

Таблица 16.2. Спецификации при вводе и выводе

Код	Описание
%d	Чтение целого числа (при выводе — переменная double)
%u	Чтение положительного целого числа (при выводе — переменная double)
%f	Чтение вещественного числа (при выводе — переменная double)
%s	Чтение строки с пробелами (при выводе — переменная типа cellstr)
%q	Чтение строки (при выводе — переменная типа cellstr)
%c	Чтение символов, разделенных пробелами (при выводе — переменная типа cellstr)

Теперь дадим пример считывания разнородных данных при помощи команды textread. Пусть в файле my.dat данные размещены на двух строках следующим образом:

```
One 1.2 -3 5
Two 1e-6 123 West
```

Кроме того, пусть в файле my.dat имеется третья пустая строка.

Напомним, что для чтения вещественной, целой и строковой переменных следует использовать соответственно спецификации %f, %d, %s. Выполним команду

```
» [nam,x,n,mar]=textread("my.dat",'%s %f %d %s');
```

После считывания данных выведем переменные nam, x, n и mar:

```
nam =
    "One"
    "Two"
x =
    1.2000
    0.0000
n =
    -3
   123
mar =
    "5"
    "West"
```

Размерности массивов nam, x, n, mar заранее не задаются, а определяются в процессе чтения по числу строк. Заметим, что при чтении вещественной переменной из второй строки никакого усечения не произошло, в чем можно убедиться, обратившись к этому элементу массива:

```
» x(2)
ans =
  1.0000e-006
```

Для считывания того же файла могут быть использованы другие команды:

```
» fid=fopen("my.dat");
while 1
                                    line = fgetl(fid)
                                    if ~isstr(line), break, end
end
fclose(fid);
```

В результате получим набор текстовых строк:

```
line =
One 1.2 -3 5
line =
Two 1e-6 123 West
line =
   Empty string: 1-by-0
line =
    -1
```

Предпоследняя строка появилась из-за того, что в считываемом файле имелась пустая строка, а последняя строка есть реакция команды fgetl на достигнутый конец файла.

Команды для работы со стандартными файлами

О сохранении и считывании рисунков в различных графических форматах написано в главе 14 «Графика MATLAB». Для считывания аудио- и видеофайлов в стандартах au, wav и avi имеются соответственно команды auread, wavread и aviread. Аналогично для записи используются команды: auwrite, wavwrite и avifile. Это позволяет применять MATLAB для обработки и подготовки звуковой и видеоинформации.

В MATLAB поддерживается работа с данными электронных таблиц Lotus 123 и Excel. Для считывания и записи файлов в стандарте Lotus 123 используются команды wk1read и wk1write, а для считывания таблиц Excel имеется команда xlsread.

Кроме того, в MATLAB существует возможность работы с данными в формате hdf (Hierarchical Data Format). Этот машинно-независимый формат разработан для хранения научных данных. Краткие описания некоторых перечисленных типов файлов даны в третьей части книги.

Объектно-ориентированное программирование

Современное объектное программирование основано на введении новых типов данных (классов) и определении операций для них. В системе MATLAB существу-

ют следующие стандартные классы: числовые массивы (double), двумерные разреженные матрицы (sparse), строки или массивы символов (char), структуры или массивы записей (struct), массивы ячеек (cell). Кроме того, некоторые расширения MATLAB используют свои классы, см., например, главу 17 «Расширения MATLAB», где описан Symbolic Math Toolbox, использующий класс объектов sym.

В MATLAB тип переменной или соответствующий класс не объявляется и не описывается, а определяется автоматически при создании объекта. Например, при задании матрицы A=eye(5) создается объект класса double, а по команде As=sparse(A) производится преобразование матрицы A в разреженную матрицу As, то есть создается новый объект класса sparse.

Объекты нового класса создаются при помощи специальной m-функции — конструктора класса, имя которой совпадает с названием класса. Для нового класса также нужно определить методы обработки объектов, написав соответствующие функции. Конструктор класса и функции обработки должны размещаться в каталоге, имя которого начинается с символа «@», а далее идет имя класса. Каталог класса является подкаталогом для каталога, описанного в пути доступа. Объекты класса реализуются в виде структур, поля которых видны только для методов, работающих с данным классом. При этом возможно доопределение существующих встроенных операторов, например, чтобы задать операцию умножения для объектов нового класса, в каталоге класса следует написать функцию mtimes.

Так, в каталоге symbolic находятся три подкаталога: @char, @sym, @symlibs. В подкаталоге @sym содержится конструктор класса — файл sym.m и множество других файлов, реализующих операции с символьными переменными. Например, для представления символьных объектов имеется функция display:

```
function display(X)
%DISPLAY Display function for syms.

%   Copyright (c) 1993-98 by The MathWorks, Inc.
%   $Revision: 1.12 $  $Date: 1997/11/29 01:05:32 $

if isequal(get(0,'FormatSpacing'),'compact')
   disp([inputname(1) " ="]);
   disp(X)
else
   disp(" ")
   disp([inputname(1) " ="]);
   disp(" ");
   disp(X)
end
```

При вызове конструктора динамически создается объект. Если конструктор вызывается без аргументов, то создается шаблон объекта, обычно с пустыми полями. Если на вход конструктора подается объект того же класса, то конструктор обычно возвращает сам введенный объект. Для проверки принадлежности объекта OBJ классу CLASS служит команда

```
isa(OBJ,CLASS)
```

Функция class (напомним, что MATLAB различает большие и малые буквы) определяет класс объекта при вызове class(OBJ) и переводит структуру S в объект класса CLASS при обращении class(S,CLASS).

В табл. 16.3 перечислены команды для работы с классами.

Таблица 16.3. Команды работы с классами

Команда	Назначение
class	Определение класса или создание объекта
isa	Определение принадлежности объекта к классу
isobject	Выявление принадлежности объекта к какому-нибудь классу
methods	Вывод списка методов для данного класса
inferiorto	Указание подчиненности класса по отношению к некоторым классам
superiorto	Указание старшинства класса по отношению к некоторым классам

При работе с объектами учитываются иерархия (старшинство) и наследование, когда объекты одного класса приобретают свойства других классов. При простом наследовании (один класс-родитель) дочерний объект включает все поля родительского объекта, и при работе с ним можно обращаться к методам родительского класса, кроме того, добавляются специфические для дочернего класса поля. При этом методы нового класса не могут применяться к объектам родительского класса. Для реализации множественного наследования (несколько классов родителей) применяется команда class(S,CLASS,PARENT1,PARENT2,...), здесь из структуры S создается объект класса CLASS, наследующий свойства классов PARENT1, PARENT2, ...

В дополнение к наследованию поддерживается объединение частей в одно целое или агрегирование, то есть использование объектов в качестве полей для других объектов.

Рассмотрим пример создания класса quaternion для работы с кватернионами, которые можно рассматривать как обобщение комплексных чисел, см., например, [МЭ]. Конечно, сегодня кватернионы не очень популярная тема, но их создатель У. Гамильтон помимо кватернионов ввел в оборот также векторы и тензоры, а в 1895 г. был даже основан «Всемирный союз в поддержку кватернионов» [К89].

В каталоге \work организуем подкаталог @quaternion и приготовим конструктор класса — функцию quaternion.m:

```
function q=quaternion(x,y,z,w)
%Quaternion x+iy+jz+kw class constructor
if nargin==0,    q.x=0; q.y=0; q.z=0; q.w=0;
elseif nargin==1,
   if isa(x,'quaternion'), q=x;
   elseif isa(x,'double'), q.x=x(1);
      leng=length(x);
      if leng==1, q.y=0; q.z=0; q.w=0;
      elseif leng==2, q.y=x(2); q.z=0; q.w=0;
      elseif leng==3, q.y=x(2); q.z=x(3); q.w=0;
      else q.y=x(2); q.z=x(3); q.w=x(4);
      end
   end
elseif nargin==2, q.x=x(1); q.y=y(1); q.z=0; q.w=0;
elseif nargin==3, q.x=x(1); q.y=y(1); q.z=z(1); q.w=0;
```

```
else   q.x=x(1);q.y=y(1); q.z=z(1); q.w=w(1);
end
q=class(q,'quaternion');
```

Приводимые далее функции имеют демонстрационный характер, так что не все случаи обработки вызовов описаны полностью. В частности, для функции quaternion.m обработка вызова с двумя и тремя входными параметрами упрощена.

Теперь приготовим функции для работы с объектами класса. Чтобы использовать для объектов нового класса знаки основных математических операций, создадим в каталоге @quaternion функции с именами plus и mtimes, которые отвечают, соответственно, операциям сложения и матричного умножения. Список имен математических операций можно получить по команде help *.

Реализуем операцию сложения кватернионов при помощи метода plus:

```
function q=plus(a,b)
%Quaternion plus
q.x=a.x+b.x; q.y=a.y+b.y; q.z=a.z+b.z; q.w=a.w+b.w;
q=class(q,'quaternion');
```

Аналогично для умножения кватернионов напишем функцию:

```
function q=mtimes(a,b)
%Quaternion plus
q.x=a.x*b.x-a.y*b.y-a.z*b.z-a.w*b.w;
q.y=a.y*b.x+a.x*b.y+a.z*b.w-a.w*b.z;
q.z=a.z*b.x+a.x*b.z+a.w*b.y-a.y*b.w;
q.w=a.w*b.x+a.x*b.w+a.y*b.z-a.z*b.y;
q=class(q,'quaternion');
```

Для преобразования кватерниона к текстовому виду создадим функцию char.m, реализующую метод представления переменных класса quaternion:

```
function s=char(q)
%Quaternion char
s=[num2str(q.x) "+I*" num2str(q.y) "+J*" ...
   num2str(q.z) "+K*" num2str(q.w)];
```

Чтобы просматривать объекты, желательно иметь специальный метод display. Для этого приготовим функцию display.m, которая будет выводить содержимое объекта, если в командной строке выражение с переменной этого класса не завершается разделителем точкой с запятой:

```
function display(q)
%Quaternion display
disp([inputname(1) "="]); disp(char(q));
```

Итак, в результате в каталоге @quaternion должны появиться функции с именами char.m, display.m, mtimes.m, plus.m и quaternion.m.

Приведем несколько примеров обращения к построенным функциям. Определим кватернион по двум скалярным величинам:

```
» q1=quaternion(0,1)
q1=
0+I*1+J*0+K*0
```

Используем в качестве входного параметра вектор

```
» q2=quaternion([1 2 3 4])
q2=
1+I*2+J*3+K*4
```

Вычислим выражение с использованием доопределенных для класса quaternion операций умножения и сложения:

```
» q3=q1*q2+q1
q3=
-2+I*2+J*-4+K*3
```

Более детальные примеры по объектно-ориентированному программированию в среде MATLAB можно найти в документации по пакету или книгах [17, 20].

Отладка

«Так уж человеку на роду написано — ошибаться до самой смерти», — справедливо замечал бравый солдат Швейк. Если в процессе выполнения программы (m-функции) возникает ошибка, то MATLAB выводит диагностическое сообщение и номер строки, в которой, по мнению пакета, произошла ошибка. Эта информация полезна, но может быть довольно приблизительной, поэтому желательно в процессе отладки программы иметь доступ к переменным для контроля вычислений и просмотра данных.

Для отладки программ и m-функций могут быть полезны следующие простые рекомендации:

- Удалите подавляющие вывод данных на экран точки с запятой или замените их запятыми. Тогда анализ промежуточных результатов может указать причину неверной работы.
- Расставьте по тестируемому тексту операторы keyboard. При выполнении этих операторов MATLAB приостановит работу, выведет приглашение (K»), и в режиме интерпретатора можно будет анализировать промежуточные результаты (выводить содержимое интересующих переменных, переопределять величины).
- Превратите m–функцию в файл script. Закомментируйте описания в начале файла и выполните полученный файл.

Кроме того, в MATLAB имеются специальные возможности для тестирования программ: набор команд отладки и средства редактора-отладчика medit. В этом случае отладка заключается в расстановке точек останова, прогоне программы от точки к точке и просмотре данных. Командный режим отладки является универсальным, поскольку одинаков для различных вычислительных систем, но для среды Windows режим отладки с использованием редактора medit выглядит нагляднее и проще.

Отладка в командном режиме

Если отладку проводить при помощи команд среды MATLAB, то для установки и снятия точек останова применяются команды dbstop и dbclear. Каждая из этих

команд имеет несколько вариантов применения, в частности команда dbstop еще и задает режимы отладки, а dbclear отменяет назначения. Список команд отладки приведен в табл. 16.4.

Таблица 16.4. Команды отладки

Команда	Описание
dbstop	Определить точку останова
dbclear	Удалить точку (точки) останова
dbcont	Продолжить выполнение
dbstep	Выполнить команду отладки
dbquit	Завершить отладку
dbstack	Список точек останова

Полный список команд отладки с перечислением вариантов обращения можно посмотреть в системе справки (help debug), документации MATLAB или книгах, см., например, [П99].

Рассмотрим отладку для функции вычисления правой части системы Лоренца. Чтобы определить точку останова в тексте функции lor, выполним команду dbstop и укажем номер строки, по достижении которой работа функции должна быть приостановлена:

```
» dbstop lor 1
```

Запустим функцию lor

```
» lor(0,[0 1 5]','',10,25,8/3)
```

Когда интерпретатор достигнет помеченной строки, то выполнение функции остановится и MATLAB укажет соответствующее место (для Windows будет запущен редактор medit и выделена строка), а в командном окне появится приглашение вводить команды в отладочном режиме K». Используя это приглашение, можно вводить любые команды, например выведем значение параметра b:

```
K» b
b =
    2.6667
```

Для продвижения отладки на шаг наберем

```
K» dbstep
```

Чтобы выйти из отлаживаемой функции, используем команду dbup, а чтобы вернуться — dbdown. Посмотреть содержимое нужной функции можно при помощи dbtype, а стек вызова функций будет выведен по команде dbstack. Для завершения отладки введем

```
K» dbquit
```

Чтобы убрать точки останова в функции lor, выполним команду

```
» dbclear lor
```

Редактор medit и отладка

Редактор-отладчик medit обеспечивает удобную среду для подготовки и отладки m-файлов в MATLAB. Файл редактора medit.exe находится в директории \bin и может быть вызван из командного окна MATLAB (пункты New и Open меню File, запуск команды edit) или запущен как обычная программа. Редактор-отладчик предоставляет доступ к текстам и данным, в нем можно редактировать m-файлы, вызывать для просмотра и изменения матричные переменные, организовывать отладку программ и функций.

Для просмотра данных из рабочей области следует вызвать Workspace Browser и выбрать (щелкнуть мышкой) нужную переменную. В редакторе откроется новый документ и появится таблица (матрица) с данными, которые можно просмотреть и переопределить. Эта возможность пока предоставляется для скалярных, векторных и матричных величин, так что многомерные массивы и структуры придется просматривать, запуская из командной строки вывод самой переменной или какой-то ее части.

Если редактор medit вызван из командного окна, то в процессе вычислений редактирование становится невозможным. Чтобы в среде Windows подготавливать код (файлы сценариев и функций) параллельно вычислениям, можно запустить редактор medit автономно от MATLAB, но в этом случае просмотр данных будет невозможен.

Рассмотрим отладку с использованием средств редактора-отладчика. Назначения значков и соответствующих им пунктов меню Debug приведены в табл. 16.5.

Таблица 16.5. Пункты меню и значки отладки редактора medit

Значки	Пункт меню	Описание
	Set/Clear BreakPoint F12	Расставить/убрать точку останова
	Clear All BreakPoints	Убрать все точки останова
	Step In F11	Построчное исполнение с заходом в вызываемые функции
	Single Step F10	Построчное исполнение
	Continue	Выполнение до следующей точки останова
	Quit Debugging	Завершить отладку

Ряд пунктов меню позволяет установить режимы отладки, предусматривающие остановку при возникновении определенных ситуаций, см. табл. 16.6.

Для отладки m-функции вызовем из командного окна редактор medit и считаем нужный файл. Чтобы указать точку останова в тексте функции, установим курсор на нужной строке и выполним пункт Set/Clear BreakPoint или щелкнем по отвечающему этому пункту значку. После расстановки точек останова запустим отлаживаемый файл. Пакет MATLAB выполнит все команды до первой точки и сделает

паузу. Здесь можно просмотреть значения переменных, а затем продолжить работу программы.

Таблица 16.6. Пункты-указатели из меню отладки Debug

Пункт меню	Назначение
Stop if Error	Остановка при ошибке
Stop if Warning	Остановка при предупреждающем сообщении
Stop if NaN or Inf	Остановка при появлении нечисловой или бесконечной переменной

На рис. 16.1 и 16.2 отражен процесс отладки рассмотренной в главе 15 «Численный анализ в MATLAB» функции lor для вычисления правой части системы Лоренца.

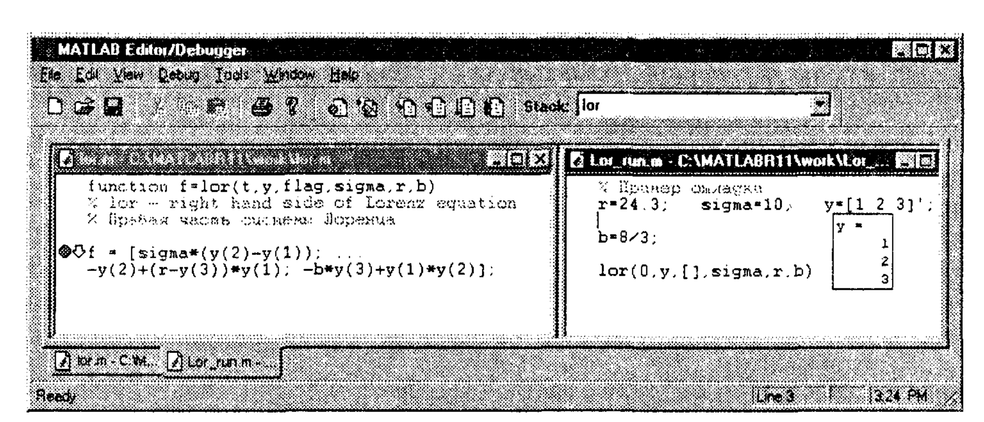


Рис. 16.1. Окно редактора-отладчика medit (2 файла)

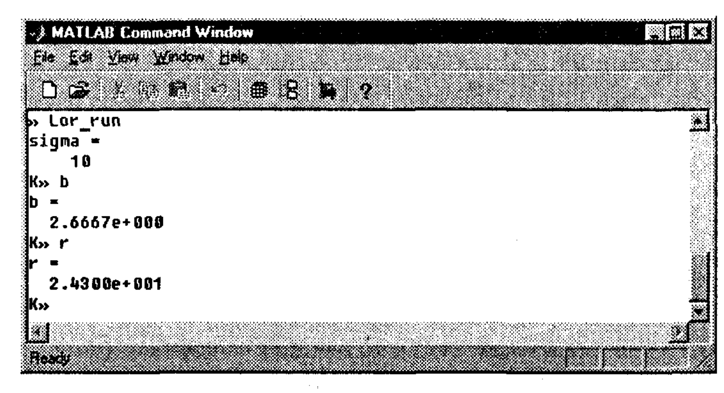


Рис. 16.2. Командное окно

На рис. 16.1 дано окно редактора medit с m-функцией lor.m и файлом сценария Lor_run.m, содержащим команды ввода параметров и обращение к функции lor. Окно с файлом lor.m находится слева, в нем точка останова помечена кружком. Эта точка пройдена, на что указывает стрелка «вниз» на строке файла lor.m, но режим отладки пока не переместился в файл Lor_run.m, так как выделена закладка с име-

нем файла lor.m в строке состояния (нижняя часть окна medit) и в окне Stack стоит имя lor. Обратим также внимание на прямоугольник с содержимым переменной y (вектора-столбца) в подокне с текстом файла Lor_run.m. В процессе отладки можно узнать компоненты любой матричной переменной, наведя мышь на идентификатор этой переменной в окне редактора medit.

На рис. 16.2 приведено окно MATLAB, содержащее вызов файла сценария Lor_run.m, результат задания переменной sigma и вывод отладочных значений переменных b и r. В режиме отладки приглашение командной строки имеет вид K» и содержимое любой переменной может быть выведено простым указанием нужного имени и нажатием клавиши Enter. На рис. 16.2 было выведено значение переменной sigma, так как вывод на экран не был подавлен в тексте файла Lor_run.m (операция присваивания закончилась запятой).

Эффективность программ и профилер m-файлов

При разработке достаточно больших программ, состоящих из нескольких функций, возникает необходимость оптимизации кода. Для этого важно знать, выполнение каких функций, а также каких их участков требует наибольших затрат времени. Здесь могут быть использованы счетчик числа операций, команды, оперирующие со временем, и, наконец, профилер m-файлов. Простейший способ оценить объем вычислений предоставляет команда flops, обращение к которой без параметров выводит число выполненных операций, а вызов flops(0) обнуляет счетчик числа операций. Это довольно грубый способ оценки быстродействия, что показывает следующий пример:

```
» flops(0), for k=1:7^7, sin(k); end, flops
ans =
     823545
```

Тот же результат будет получен, если вычисление синуса заменить сложением целых чисел:

```
» flops(0), for k=1:7^7, k+1; end, flops
ans =
     823545
```

Имеющиеся в MATLAB команды работы со временем и датами позволяют оценить время расчета отдельных фрагментов кода. Команда tic обнуляет время, а toc замеряет прошедшее с момента выполнения tic время. Применим эти команды для оценки времени выполнения двух циклов, различающихся тем, что в теле первого вычисляется синус переменной цикла, тогда как второй цикл просто прибавляет единицу к переменной цикла:

```
» tic, for k=1:7^7, sin(k);end, toc
elapsed_time =
  9.6700e+000
» tic, for k=1:7^7, k+1;end, toc
```

```
elapsed_time =
   8.1300e+000
```

Команда cputime подсчитывает процессорное время с момента запуска сеанса. Кроме того, имеются команды clock, etime, date для получения соответственно вектора-строки с данными (год, месяц, день, час, минуты, секунды), определения длительности временного промежутка между двумя моментами времени и вывода символьной строки (день–месяц–год).

Чтобы повысить эффективность вычислений, нужно знать, какие операции какое время расходуют. Для этого в MATLAB имеется программа профилер. По умолчанию профилер собирает информацию по m- и mex-файлам, а также подфункциям. Рассмотрим работу профилера на примере анализа времени решения системы Лоренца, см. главу 15 «Численный анализ в MATLAB».

Запустим профилер с параметром -detail builtin, чтобы собирать статистику также по встроенным (builtin) функциям:

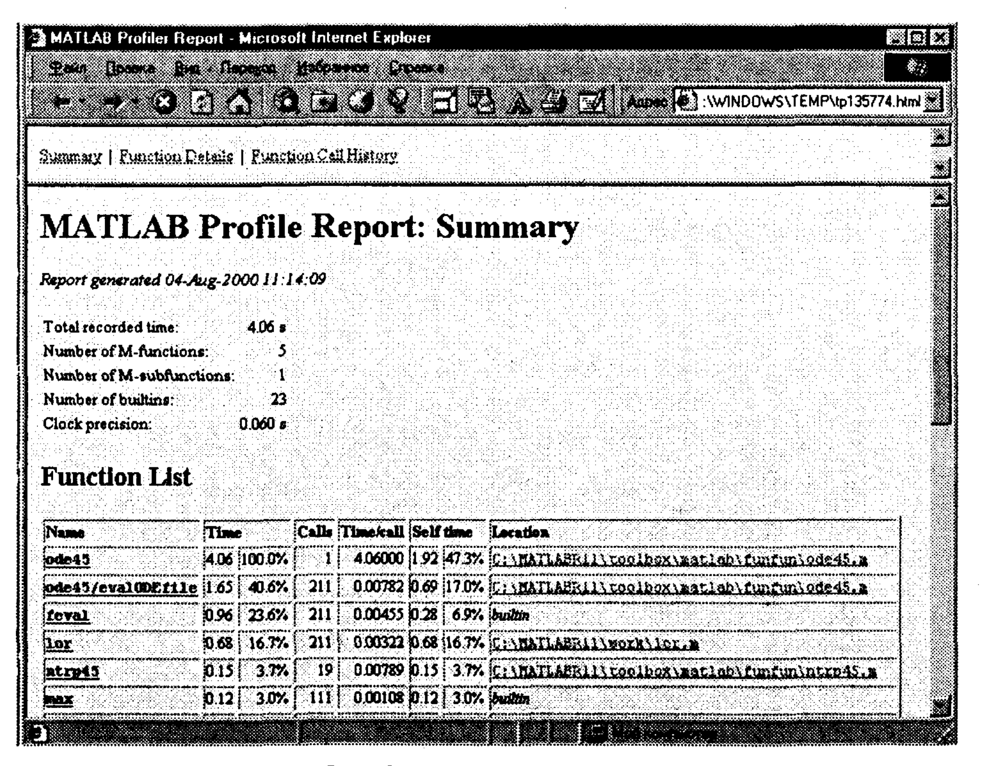
```
» profile on -detail builtin -history
```

Затем выполним m-файл:

```
» [T,Y]=ode45("lor",[0 5],[0 1 1],[],10,25,8/3);
```

Сгенерируем отчет:

```
» profile report
```



MATLAB Profile Report: Summary

Report generated 04-Aug-2000 11:14:09

Total recorded time:	4.06 s
Number of M-functions:	5
Number of M-subfunctions:	1
Number of builtins:	23
Clock precision:	0.060 s

Function List

Name	Time		Calls	Time/call	Self time		Location
ode45	4.06	100.0%	1	4.06000	1.92	47.3%	C:\MATLABR11\toolbox\matlab\funfun\ode45.m
ode45/evalODEfile	1.65	40.6%	211	0.00782	0.69	17.0%	C:\MATLABR11\toolbox\matlab\funfun\ode45.m
feval	0.96	23.6%	211	0.00455	0.28	6.9%	*builtin*
lor	0.68	16.7%	211	0.00322	0.68	16.7%	C:\MATLABR11\work\lor.m
ntrp45	0.15	3.7%	19	0.00789	0.15	3.7%	C:\MATLABR11\toolbox\matlab\funfun\ntrp45.m
max	0.12	3.0%	111	0.00108	0.12	3.0%	*builtin*

Рис. 16.3. Начало отчета профилера

При этом в окне Интернет-браузера появится HTML-файл с отчетом, см. рис. 16.3.

На рис. 16.3 приведено гипертекстовое окно с результатами работы профилера. Пользуясь гиперссылками, можно выяснить подробности проведенного анализа (число обращений к функциям, потраченное время и т. д.). Например, можно получить данные о затраченном времени, числе обращений, среднем времени на обращение и другую информацию для функции lor, см. рис.16.4.
Вновь запустим профилер без очистки полученной статистики:

```
» profile resume
```

Профилер готов продолжать собирать данные по другим функциям. Новые данные будут добавлены к полученным ранее.

Сбор статистики завершается, и работа профилера прекращается по команде

```
» profile off
```

Распечатать результат можно, отправив на печать файл из браузера или выполнив команду

```
» profile report NAME
```

Здесь NAME — имя HTML-файла, в котором будет сохранена статистика, собранная профилером. В версии MATLAB 6 команды профилера остались такими же, как в версии 5.3.

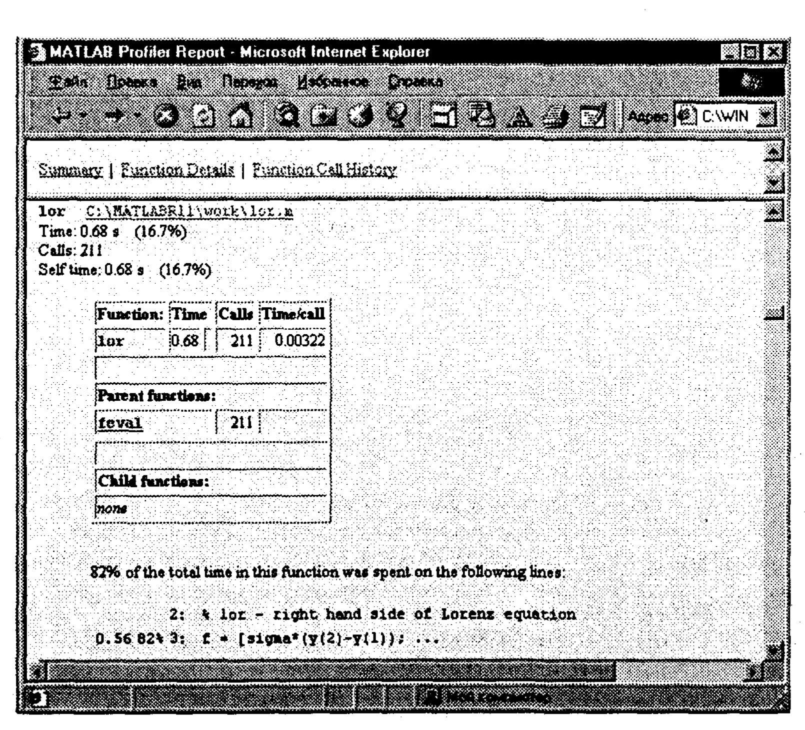


Рис. 16.4. Фрагмент отчета профилера

Разработка mex-файлов

Для ускорения вычислений в MATLAB можно использовать mex-файлы — написанные на С или Фортране и откомпилированные процедуры. Тексты на языке MATLAB (файлы с расширением .m) машинно-независимы, а подготовленные для одной платформы mex-файлы (Matlab EXternal) должны быть откомпилированы заново при переходе к другой среде. Для каждой из платформ подготовленные mex-файлы имеют свое расширение, например: dll для Windows, mexlx для Linux, mexsol для Solaris, mexalp для Alpha.

Вызываются mex-файлы как обычные m-файлы, причем если интерпретатору встретилось некоторое имя функции, то вначале ищется mex-файл с этим именем в каталогах, определенных списком MATLAB search path, и только при его отсутствии начинается поиск m-файла.

Программирование mex-файлов заключается в написании процедур на С или Фортране и подготовке переходного блока для получения данных из среды MATLAB и возвращения результата. MATLAB оперирует с массивами, поэтому все переменные — скаляры, векторы, матрицы, строки, массивы ячеек, структуры и объекты — реализованы в виде массивов. В программе на С массив MATLAB должен быть объявлен типом mxArray, в нем будут содержаться размерности, сами данные, сведения об их типе (вещественные или комплексные), индексы ненулевых элементов для разреженных (sparse) матриц, число полей и их имена для структуры и объекта. Заметим, что из m-файла при помощи имеющегося в MATLAB компилятора lcc можно приготовить текст на языке С, что обсуждается в следующем разделе.

При подготовке mex-файлов следует определить используемый компилятор, для чего надо выполнить команду

```
» mex -setup
```

Если на машине установлено несколько компиляторов, то MATLAB предложит выбрать, какой из них будет использоваться далее для компиляции. Например, если на машине установлен только компилятор Borland C/C++, то будет предложен следующий выбор:

```
Select a compiler:
[1] Borland C/C++ version 5.02 in C:\BC5
[2] Lcc C version 2.4 in C:\MATLABR12\sys\lcc

[0] None

Compiler:
```

Здесь lcc — собственный компилятор MATLAB. Для компиляции текстов на Фортране также следует определить соответствующий компилятор.

В комплекте поставки MATLAB имеются примеры оформления mex-файлов. Так, в подкаталоге \extern\examples\mex находятся файлы на С (yprime.c) и Фортране (yprimef.f, yprimefg.f). Чтобы запустить компиляцию примера на С, надо перейти в подкаталог с примером и вызвать команду mex. Например, чтобы откомпилировать функцию yprime.c, выполним следующие команды:

```
cd([matlabroot "\extern\examples\mex"])
mex yprime.c
```

Взаимодействие интерпретатора с mex-файлом осуществляется следующим образом. Пусть func — m-функция, которая должна быть реализована в виде mex-файла. Обращение

```
[C,D]=func(A,B)
```

указывает MATLAB, что параметры A, B являются входными, а параметры C, D — выходными. Заголовок принимающей функции на языке С должен иметь имя mexFunction и оформляться следующим образом:

```
#include <math.h>
void mexFunction(
                    int nlhs, mxArray *plhs[],
                    int nrhs, const mxArray *prhs[]
)
```

Переменные plhs и prhs есть векторы, содержащие указатели на выходные и входные параметры. Чтобы воспринять входные данные, следует обратиться к функции mxGet и использовать указатели prhs[0], prhs[1] и т. д. В случае скалярной переменной можно работать непосредственно с ее значением, получить которое позволяет функция mxGetScalar. Для оформления возвращаемых параметров нужно использовать функцию mxCreate.

Для компиляции mex-функции следует запустить команду mex. При обнаружении компилятором ошибки будет выведено диагностическое сообщение и указан номер ошибочной строки. В случае нормального завершения компиляции появится файл с именем исходного файла и расширением dll.

Приведем пример создания mex-файла. Перепишем функцию вычисления правой части системы Лоренца на языке С. Поскольку система автономна, то опустим переменную t и служебную строковую переменную flag.

Подготовленный файл lore.c имеет вид:

```
#include "mex.h"
/* lore - right hand side of Lorenz equation */
void lore(double f[], double y[],
double sigma, double r, double b)
{
                        f[0]= sigma*(y[1]-y[0]);
                        f[1]= -y[1]+(r-y[2])*y[0];
                        f[2]= -b*y[2]+y[0]*y[1];
}
void mexFunction( int nlhs, mxArray *plhs[],
int nrhs, const mxArray *prhs[] )
{
double *f, *y, sigma, r, b;
int mro, nco;
/* Проверка аргументов   */
                        if (nlhs>1) {mexErrMsgTxt("Много выходных данных");}
/* Создаем выходной массив      */
                        mro=mxGetM(prhs[0]);  nco = mxGetN(prhs[0]);
                        plhs[0]=mxCreateDoubleMatrix(mro,nco,mxREAL);
```

```
/* Назначим указатель на выходной массив */
                              f=mxGetPr(plhs[0]);
/* Назначим указатель на входной массив */
                              y=mxGetPr(prhs[0]);
/* Определим скалярные входные параметры */
                              sigma=mxGetScalar(prhs[1]);
                              r=mxGetScalar(prhs[2]);
                              b=mxGetScalar(prhs[3]);
/* Вызов функции          */
                              lore(f,y,sigma,r,b);
}
```

Запустим компиляцию подготовленного файла:

```
mex lore.c
```

При нормальном завершении компиляции появится файл lore.dll. Проверим, верно ли работает откомпилированная функция, для чего с одинаковыми входными данными обратимся к функции `lor`, заданной файлом lor.m, и к подготовленной функции lore. В результате вызова функции `lor` получим:

```
» lor(0,[0 1 5]','',10,25,8/3)'
ans =
   10.0000   -1.0000  -13.3333
```

Такой же результат будет получен для `lore`:

```
» lore([0 1 5]',10,25,8/3)'
ans =
   10.0000   -1.0000  -13.3333
```

Использовать mex-функцию `lore` для интегрирования системы Лоренца нельзя, так как перечень ее входных параметров не отвечает набору, требуемому интеграторами `ode23`, `ode45` и др. Поэтому дополнительно напишем функцию-переходник (обычный m-файл) с единственной целью — вызывать mex-функцию для вычисления правой части системы Лоренца:

```
function f=lorenz(t,y,flag,sigma,r,b)
% lor - right hand side of Lorenz equation
f = lore(y,sigma,r,b);
```

Теперь можно использовать подготовленные функции `lore` и `lorenz` для интегрирования системы Лоренца.

При выполнении mex-задания используются параметры, перечисленные в файле параметров `mexopts`, например mexopts.bat для Windows. Параметры можно также указывать в командной строке. В табл. 16.7 перечислены некоторые параметры для команды `mex`, доступные для разных платформ.

В рассмотренном примере функции, начинающиеся с приставки `mex`, были функциями MATLAB. Из mex-функций можно обращаться и к другим командам MATLAB; для этого имеется специальная функция `mexCallMATLAB` с пятью параметрами. Первые четыре обеспечивают интерфейс и аналогичны параметрам функции `mexFunction`, а пятый содержит имя вызываемой команды.

В среде Windows скомпилированный машинный код mex-файла находится в файле динамической библиотеки (Dynamic Link Library) с расширением .dll. Для отладки mex-функций можно использовать среду системы, в которой компилируется функция, например Developer Studio для Visual C++ Microsoft.

Таблица 16.7. Параметры команды mex

Параметр	*Назначение*
-c	Компиляция без редактирования связей
-h	Справка по команде mex
-g	Включение отладочных символов в mex-функцию
-I PATH	Указание пути (PATH) для компилятора
-O	Оптимизация mex-функции
-output NAME	Указание имени (NAME) создаваемой mex-функции
-v	Печать установок компилятора и редактора связей

Более подробно программирование и отладка mex-функций описаны в фирменном руководстве Application Program Interface Guide, см. также [20].

Компилятор MATLAB

При помощи MATLAB можно создавать независимые приложения, используя компиляторы C и C++. Для этого предназначены компилятор MATLAB Compiler и библиотека функций C++ Math Library. Достоинством такого подхода является гибкость и простота применения: синтаксис аналогичен используемому в MATLAB, все арифметические вычисления ведутся с двойной точностью, за счет исключения этапа интерпретации возрастает скорость выполнения функций, библиотечный код сам управляет выделением памяти и ее очисткой.

Библиотека C++ Math Library состоит из интерфейсных функций, набора бинарных и унарных операторов и собственно математических функций MATLAB. Примерно триста встроенных и компилируемых (m-файлы) функций могут быть использованы для создания приложений на языке C. Это элементарные и специальные математические функции, наборы стандартных и специальных матриц, функции для решения задач линейной алгебры, интерполяции, статистики и анализа данных, команды ввода-вывода и др. Библиотека не поддерживает графику, команды системы Simulink, работу с разреженными матрицами и использование интерпретаторов командной строки (функций eval и feval).

Компилятор MATLAB Compiler Version 2.0 (файл mcc.m и ряд сопутствующих файлов) позволяет конвертировать m-файл в файл на языке C или C++, а также организует поддержку бинарных файлов. По умолчанию результаты компиляции записываются в директорию \work. Драйвер с именем mcc служит для запуска компиляции, для получения кода на C имеется специальный mex-файл mccexec, а для построения mex-файлов и отдельных приложений — mbuild.

Обращение к компилятору для трансляции файлов fun, fun2 с параметрами options имеет вид

```
mcc [options] fun [fun2 ... ]
```

В квадратных скобках указаны необязательные параметры. Описание всех функций и набор примеров можно получить по справке help mcc. Для ряда наиболее

важных задач, решаемых с помощью компилятора MATLAB Compiler, имеются макропараметры — подготовленные комплекты из основных параметров. Перечни макро- и основных параметров даны соответственно в табл. 16.8 и 16.9. Например, если файл fun.m содержит функцию на m-языке, то для запуска компилятора для перевода (трансляции) этой функции в файл на языке C++ и генерации приложения достаточно команды с одним макропараметром -p:

```
mcc -p fun
```

Эквивалент, использующий основные опции, выглядит следующим образом:

```
mcc -t -W main -L Cpp -h -T link:exe fun
```

Таблица 16.8. Макропараметры компилятора

Параметр	Назначение
-x	Генерация файла на языке C и mex-файла
-m	Генерация файла на языке C и приложения
-p	Генерация файла на языке C++ и приложения
-S	Генерация Simulink-функции (s-функции) mex-файла

Таблица 16.9. Основные параметры компилятора

Параметр	Назначение
-c	Конвертация m–файла в текст на языке C
-d <directory>	Указание каталога для файла-результата
-g	Включение отладочной информации
-I <path>	Подключение путей доступа (каталогов)

Если нужно взять файл из каталога source, а результат поместить в каталог target, то обращение должно быть оформлено следующим образом:

```
mcc -m -I /source -d /target fun
```

Приведем пример использования компилятора для преобразования m-файлов в тексты на C и последующего создания выполняемого приложения из двух функций. В первой функции fun1 из файла 111.txt считывается целое число, далее организуется цикл, в котором стоит обращение к функции fun2. По завершении цикла в файл 222.txt записывается максимальное число из массива, вычисленного последним. В начале и конце функции fun1 стоят соответственно команды tic и toc, что позволяет оценить время выполнения тела функции fun1:

```
function y=fun1
tic
f1=fopen("111.txt",'r'); f2=fopen("222.txt",'w');
n = fscanf(f1,'%5d'), fclose(f1);
for k=1:n, ei=fun2(n); end
fprintf(f2,'%9.2f',max(ei)); fclose(f2);
toc
```

Функция fun2 с одним входным параметром n возвращает спектр магической матрицы порядка n:

```
function y=fun2(n)
y=eig(magic(n));
```

Приведем последовательность команд для подготовки приложения из файлов fun1.m и fun2.m, сопроводив выполняемые команды комментариями:

```
mcc -t -L C fun1                    % Результат: fun1.c
mcc -t -L C fun2                    % Результат: fun2.c
mcc -W main -L C fun1 fun2          % Результат: fun1_main.c
mcc -T compile:exe fun1.c           % Результат: fun1.obj
mcc -T compile:exe fun2.c           % Результат: fun2.obj
mcc -T compile:exe fun1_main.c      % Результат: fun1_main.obj
mcc -T link:exe fun1.obj fun2.obj fun1_main.obj
```

В результате выполнения этих команд в текущем каталоге появятся файлы на C fun1.c, fun2.c, fun1_main.c, объектные файлы fun1.obj, fun2.obj, fun1_main.obj и исполняемый файл fun1.exe. Отметим, что тексты полученных файлов на C не приводятся. При запуске m-функции fun1 в среде MATLAB получился следующий результат:

```
» fun1
n =
   55
elapsed_time =
  6.9200e+000
```

При других прогонах функции fun1 оценка затраченного времени варьировалась, составляя в среднем 6.7 секунды, выполнение же программы fun1.exe требовало примерно 6.4 секунды. Заметим, что для запуска fun1.exe нужны соответствующие библиотеки dll, в которых содержатся использованные функции, поэтому для работы с исполняемым файлом нужны соответствующие dll-библиотеки MATLAB. Конечно, рассмотрен простой пример, но и он дает представление об эффективности вычислений в самой среде, а также о возможностях компилятора и библиотек MATLAB для построения приложений.

В MATLAB имеется библиотека для подключения ядра пакета в программах на языках C и Фортран. Это позволяет вызывать из пользовательских программ, написанных на этих языках и подготовленных в соответствующих средах, математические, графические и другие команды MATLAB. В каталоге \extern\examples\engmat находится файл engwindemo.c с примером такой программы. Для подготовки приложения используем команду mex, где указан bat-файл с нужными настройками для компиляции:

```
mex -f c:\matlabR11\bin\bcc53engmatopts.bat engwindemo.c
```

Имя файла настройки зависит от применяемого транслятора, а сами файлы для различных компиляторов находятся в каталоге \bin. В данном примере использован компилятор Borland C. В результате выполнения этой команды получается exe-файл, который вызывает ядро MATLAB, формирующиеся при вызове графическое и текстовое окна совмещены на рис. 16.5.

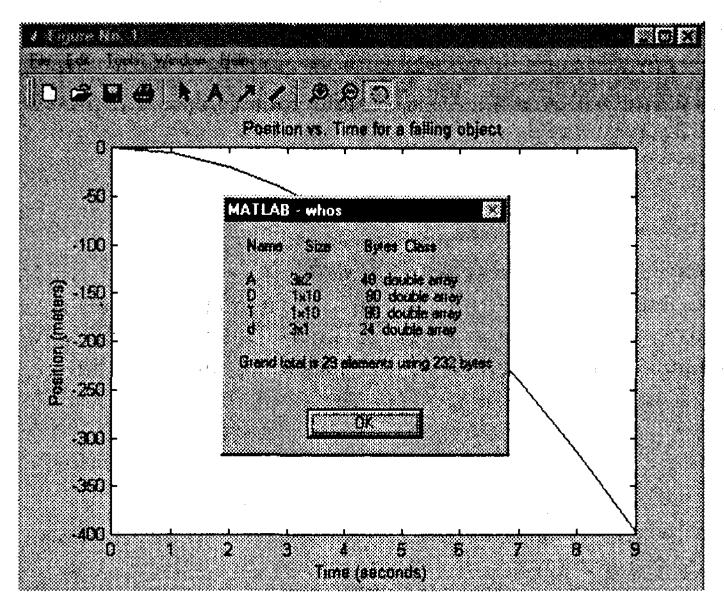


Рис. 16.5. Окна программы на C, использующей ядро MATLAB

Программирование интерфейса и организация диалога

Для взаимодействия с программой необходим интерфейс, чтобы вводить параметры задачи и управлять процессом вычислений. MATLAB имеет простые средства для создания элементов интерактивных приложений и позволяет создавать графический интерфейс, используя стандартные графические окна.

Элементарный интерфейс

Команды для реализации простого ввода, приостановки вычислений и организации несложных меню представлены в табл. 16.10.

Таблица 16.10. Команды элементарного интерфейса

Имя	Назначение
input	Интерактивный ввод
keyboard	Запрос на ввод с клавиатуры
pause	Пауза
error	Печать сообщения и выход из m-файла
menu	Меню диалогового ввода

Для организации пауз или приостановки выполнения m-файла используется команда pause. Команда pause без параметра вызывает ожидание нажатия какой-ни-

будь клавиши, а команда pause(N) организует задержку на N секунд. Команда pause off выключает режим пауз, а команда pause on восстанавливает этот режим.

Элементарный интерфейс с использованием клавиатуры можно создать при помощи команды input. Например:

```
» i=input("?")
? sin(1)
0.781
```

При вводе производится синтаксический анализ введенного текста. Подавить анализ позволяет дополнительный параметр "s", с использованием которого можно организовывать ввод чего-либо по умолчанию. Например, наберем следующую команду и на появившийся запрос просто нажмем клавишу Enter:

```
» IN=input("Новый расчет? Да/Нет [Да]:", "s")
Новый расчет? Да/Нет [Да]:
IN =
     ""
```

Теперь можно проанализировать содержимое переменной IN. Если команда isempty(IN) выведет «1», то это соответствует утвердительному ответу. Для иного ответа можно ввести что угодно:

```
» IN=input("Новый расчет? Да/Нет [Да]:", "s")
Новый расчет? Да/Нет [Да]: Вряд ли
IN =
 Вряд ли
```

Простое меню можно организовать при помощи команды menu. Приведем пример и заодно продемонстрируем, как при помощи команды keyboard организовать доступ к данным в процессе выполнения программы. В текстовом редакторе создадим файл **demo_menu.m**:

```
m=0;
while m~=3,
m=menu("Выбор",'Команда date','Команда keyboard','Выход');
switch m
case 1, today=date
case 2, keyboard
end
end
```

Запустим на выполнение файл **demo_menu.m**

```
» demo_menu
```

В результате появится меню:

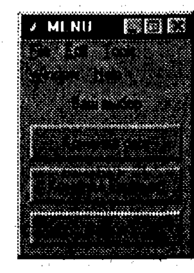


Рис. 16.6. Создание меню

При выборе первого пункта будет выведена текущая дата, при выборе второго пункта появится приглашение K» вводить команды с клавиатуры. Здесь можно посмотреть, что является текущим значением переменной m. Для выхода из режима клавиатуры наберем return и нажмем Enter. Управление снова передастся в окно с меню:

```
K» m
m =
     2
K» return
»
```

Интерфейс графических окон

Средствами MATLAB можно создавать программы, где расчет сопровождается визуализацией и для управления используется графическое меню. Для этого непосредственно в стандартном графическом окне размещаются меню и графические элементы управления, позволяющие вводить и выводить числа и текст, получать справочную информацию, производить выбор, нажатием кнопки запускать выполнение функций и т. д. Команды для создания обычного и контекстного меню, а также различных элементов управления даны в табл. 16.11.

Таблица 16.11. Команды создания графического интерфейса

Имя	Назначение
uicontrol	Создание элемента управления
uimenu	Создание меню
uicontextmenu	Создание контекстного меню

При помощи этих команд определяются объекты и задаются некоторые их свойства. Отсутствующие свойства берутся по умолчанию. Рассмотрим создание элементов управления при помощи команды uicontrol. Вначале нужно создать графическое окно и получить его дескриптор

```
hFig=figure;
```

Затем подготавливаются различные элементы управления (объекты uicontrol), создаваемые при помощи конструктора (команды) uicontrol. Первым параметром команды uicontrol идет дескриптор родительского окна, а далее в произвольном порядке перечисляются свойства, задаваемые парой: имя и значение.

Например, чтобы организовать кнопку, при нажатии которой будет вызываться функция fun (файл fun.m), достаточно ввести команду

```
hBut=uicontrol(hFig,'Style','pushbutton', ...
   "String",'Кнопка', "Position",[20 20 30 40],...
   "Callback",'fun');
```

Ключевое слово Style определяет тип управляющего элемента, здесь это кнопка **pushbutton**, а свойство **String** задает надпись на кнопке. Значением свойства **Position** будет вектор-строка из четырех чисел: первые два числа устанавливают положение кнопки относительно левого нижнего угла графического окна, третье число

задает ширину кнопки, а четвертое — ее высоту. Свойство Callback определяет имя функции fun, которая будет вызвана при нажатии данной кнопки. Файл fun.m должен быть подготовлен заранее.

Свойство Style также может принимать следующие значения:

- Text — для вывода текстовой информации;
- Edit — для ввода чисел и текста;
- Checkbox — для пометки;
- listbox — для выбора из списка;
- togglebutton — для реализации переключателя;
- radiobutton — для выбора нескольких вариантов из списка;
- slider — для реализации ползунка;
- frame — для взятия нескольких элементов в рамку.

Элементы управления могут иметь особенности. Например, элемент frame должен быть создан прежде элементов, которые заключаются в рамку. Для элемента slider должны быть указаны минимальное (слева) и максимальное (справа) значения, и дополнительно можно определить значение параметра, управляющего перемещением ползунка. При значении popupmenu появляется раскрывающееся меню.

Чтобы получить список свойств элемента управления с дескриптором hBut, достаточно выполнить команду get(hBut). Для задания нового значения свойства Prop следует использовать команду set(hBut,'Prop','Val'), где Val — значение свойства.

Приведем пример интерфейса из документации по MATLAB. Чтобы создать кнопку в центре окна и передать управление новому объекту, введем команду

```
» b=uicontrol("Style",'pushbutton','Units','normalized',...
"Position",[.6 .6 .3 .2],'Fontsize',12,'String','Кнопка');
```

Дополнительно установим всплывающую подсказку, которая будет появляться при наведении курсора на кнопку

```
» set(b,'TooltipString','Блуждающая')
```

Теперь приготовим команду, которая будет выполняться при нажатии кнопки

```
» s='set(b,''Position'',[.8*rand .9*rand .3 .2])'
```

Выполнение следующей команды приводит к случайному перемещению кнопки

```
» eval(s)
```

Наконец, запустим подготовленную программу. При каждом нажатии кнопки происходит ее перемещение на новую позицию:

```
» set(b,'Callback',s)
```

На рис. 16.7 приведено окно с кнопкой и всплывающей подсказкой.

Заметим, что в данном примере описание дескриптора окна отсутствует, а при помощи свойства Callback вместо функции успешно вызывается строковая переменная. Применение различных элементов управления рассмотрено в примере главы 10 «Примеры решения задач».

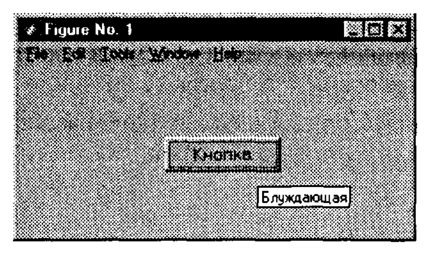


Рис. 16.7. Кнопка — пример элемента графического интерфейса

Для пополнения стандартного меню графического окна применяется команда

```
U=uimenu('Prop1',Val1, 'Prop2',Val2,...)
```

Здесь U — дескриптор создаваемого меню, Prop1, Prop2, ... — свойства, а Val1, Val2, ... — присваиваемые значения. Список свойств можно получить, выполнив команду get(uimenu).

При вызове U1=uimenu(U,...) создается дочернее меню, причем дескриптор U используется для определения характера создаваемого меню. Если U — дескриптор окна, то будет организован пункт основного меню, если U — дескриптор меню, то создастся подпункт в выпадающем меню, также могут создаваться подпункты дальнейшей вложенности и пункты контекстного меню.

Приведем простой пример. Выполним следующую последовательность команд:

```
» U=uimenu("Label",'Model');
» U1=uimenu(U,'Labe','Lorenz');U2=uimenu(U,'La','Rossler');
» U11=uimenu(U1,'L','sigma');
```

В результате выполнения этих команд организуется окно, приведенное на рис. 16.8. Заметим, что имена свойств при формировании меню могут сокращаться до тех пор, пока это не препятствует их идентификации.

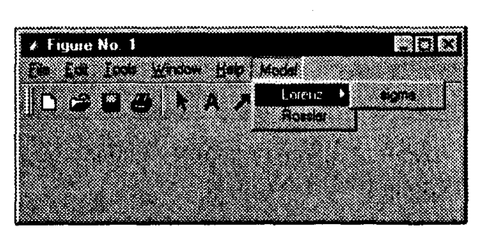


Рис. 16.8. Пример использования команды uimenu

Выведем полученные дескрипторы:

```
» [U U1 U2 U11]
ans =
  7.8001e+001  7.9000e+001  8.0000e+001  8.1000e+001
```

Знание дескрипторов необходимо для организации контекстного меню, которым сопровождается уже существующий графический объект. Команда создания контекстного меню имеет вид

```
Ui=uicontextmenu('Prop1',Val1, 'Prop2',Val2,...)
```

Здесь Ui — дескриптор создаваемого меню, Prop1, Prop2, ... — его свойства, а Val1, Val2,... — значения. Для получения списка свойств и назначений по умолчанию нужно выполнить команду get(uicontextmenu).

Воспользуемся примером с блуждающей кнопкой, разобранным выше. Создадим контекстное меню, отметив единственное свойство — видимость меню:

```
» ui=uicontextmenu("Visible",'on');
```

и организуем два пункта меню:

```
» um1=uimenu(ui,'L','Час'); um2=uimenu(ui,'L','День');
```

Наконец укажем, что контекстное меню связано с кнопкой (дескриптор кнопки b):

```
» set(b,'UIContextmenu',ui);
```

Теперь при наведении курсора на кнопку и нажатии правой кнопки мыши будет появляться контекстное меню из двух пунктов. Чтобы при выборе этих пунктов выполнялись каких-нибудь действия, следует определить свойство Callback, то есть подготовить функции, которые будут реагировать на нажатие левой кнопки мыши.

Ряд команд предоставляет интересные возможности для интерактивной работы, см. табл. 16.12.

Таблица 16.12. Команды интерактивной работы с графическими элементами

Имя	Назначение
ginput	Определение координат посредством мыши
dragrect	Перемещение прямоугольника посредством мыши
rbbox	Изменение размеров прямоугольника
selectmoveresize	Выбор, перемещение, изменение размеров
waitforbuttonpress	Ожидание нажатия клавиши или кнопки мыши
uiwait	Прекращение выполнение и ожидание команды на продолжение работы
uiresume	Возобновление выполнения

Остановимся лишь на команде интерактивного ввода координат по нажатии клавиши мыши, имеющей следующий формат:

```
[X,Y,BUT]=ginput(N)
```

Параметр N задает число вводимых точек, координаты помещаются в массивы X и Y, в массиве накапливается информация о нажатых при определении точек клавишах мыши или клавиатуры. Если параметр N отсутствует, то точки будут вводиться, пока не будет нажата клавиша Enter.

Рассмотрим простой пример. Построим график и запустим команду интерактивного ввода точек:

```
» plot([1 3 2]); hold("on"), [X,Y,Bu]=ginput;
```

Пять раз используем различные клавиши мыши и потом нажмем клавишу Enter. Затем выведем интерактивно определенные точки на график, используя кружок,

если была нажата левая клавиша мыши, и звезду — для правой клавиши. Полученный график с исходной ломаной и пятью точками

```
» ci=find(Bu==1);plot(X(ci),Y(ci),'o'),
cr=find(Bu==3);plot(X(cr),Y(cr),'*'),
```

приведен на рис. 16.9.

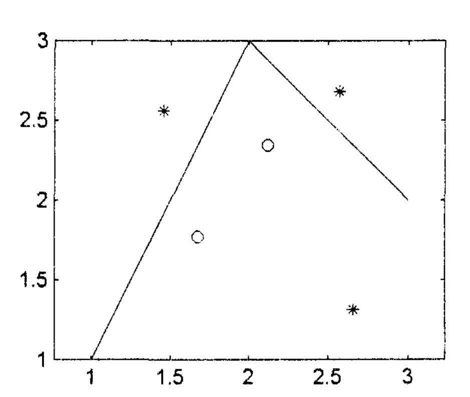


Рис. 16.9. Пример использования команды ginput

Для контроля выведем массивы координат:

```
» X=X', Y=Y', Bu=Bu'
X =
   1.4608    2.5714    1.6774    2.1198    2.6544
Y =
   2.5614    2.6784    1.7719    2.3450    1.3158
Bu =
    3     3     1     1     3
```

Интерактивная разработка графического интерфейса

Коротко остановимся на инструментах графического интерфейса пользователя GUI. Для поддержки визуальной реализации GUI в MATLAB имеется среда GUIDE (Graphical User Interface Development Environment), при помощи которой создается код на языке MATLAB. Специальный редактор `Guide Callback Editor` подготавливает код, который выполняется при нажатии клавиш.

Для формирования пользовательского интерфейса имеется специальная утилита `guide`. Ручной способ задания элементов управления, рассмотренный выше, обеспечивает максимальный контроль над процессом разработки графического интерфейса. В то же время визуальное макетирование при помощи `guide` значительно ускоряет процесс размещения и компоновки управляющих элементов. Для этого используются пять основных средств, см. табл. 16.12. В скобках даны названия соответствующих команд. Панель **Guide Control Panel** можно вызвать командой `guide` или выбрав в меню **File** командного окна пункт **Show GUI Layout Tool**.

По команде guide на дисплее появляются два окна: графическое окно-заготовка и главное окно guide (см. рис. 16.10). В нижней части окна программы guide располагается перечень графических элементов, каждый элемент имеет специфическое графическое изображение и снабжен названием. При помощи мыши можно отбуксировать нужный элемент в окно-заготовку. В средней части окна работа с элементами отражается появлением записей.

Таблица 16.13. Графические средства разработки интерфейса

Имя	Назначение (команда)
The Property Editor	Редактор свойств (propedit)
The Guide Control Panel	Панель управления (guide)
The Callback Editor	Редактор вызовов (cbedit)
The Alignment Tool	Средство выравнивания (align)
The Menu Editor	Редактор меню (menuedit)

В графическом окне организуем элементы оси (axes), кнопку (pushbutton) и рамку (frame), внутри которой поместим ползунок (slider) и текст (text), см. рис. 16.11.

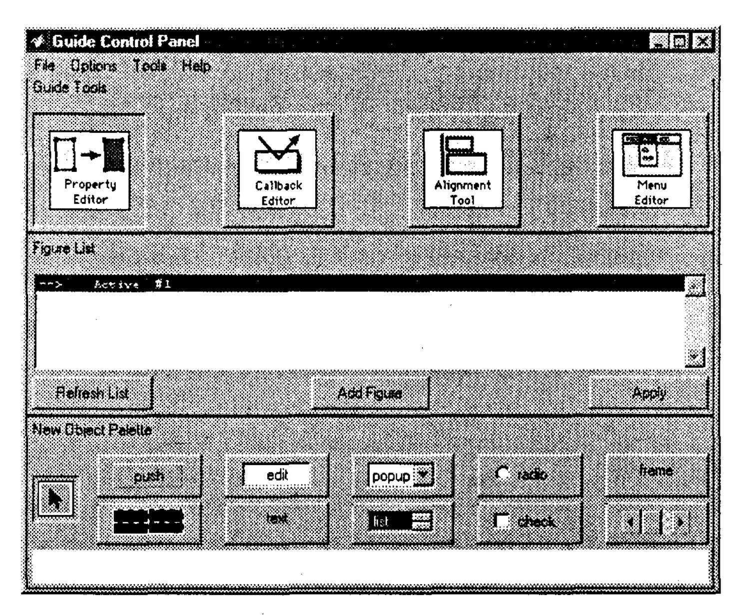


Рис. 16.10. Панель управления (The Guide Control Panel)

После размещения графических элементов вызовем редактор свойств Graphics Property Editor. Это можно сделать, выбрав пункт меню Tools графического окна, значок на панели The Guide Control Panel или дважды щелкнув мышью на выделенном элементе. В верхней части редактора свойств укажем редактируемый объект uicontrol, выберем в средней части нужное свойство и сделаем надпись над ползунком в отдельном окне. Нужные свойства теперь можно установить, выбирая в сред-

ней части свойство и редактируя его. Например, на рис. 16.12 выделено свойство 'String' и ему присвоено значение 'Скорость'.

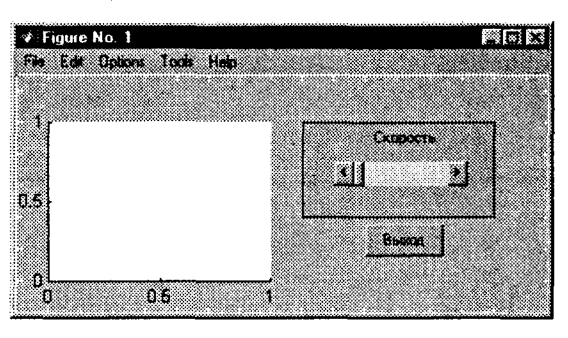


Рис. 16.11. Графическое окно

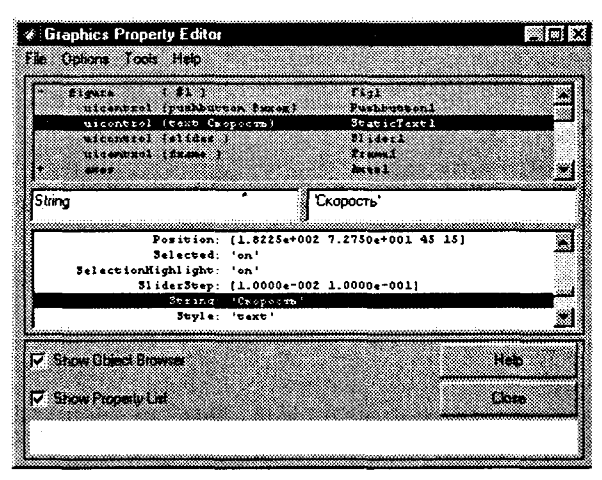


Рис. 16.12. Редактирование свойств (Graphics Property Editor)

Список свойств графического окна с дескриптором h можно получить по команде get(h). Пролистав список свойств, можно узнать значения, действующие по умолчанию. Внеся необходимые изменения и подготовив функции, которые будут выполняться при помощи вызова callback, можно закрыть утилиту guide. Перед выходом из guide последует запрос о сохранении результатов работы в виде m-файла на диске. В дальнейшем для модификации подготовленного графического окна нужно запустить этот файл на выполнение и затем вызвать команду guide.

Используя интерактивный способ оформления графического окна при помощи утилиты guide, надо иметь в виду следующее. Выравнивание элементов управления проще реализовывать вручную, задавая значения свойства Position. Может оказаться сложным найти дескрипторы для созданных элементов. Здесь можно использовать команду поиска объектов findobj, но все-таки надежнее вручную описать все дескрипторы как глобальные переменные. Наконец, если характер задачи тре-

бует перестройки элементов управления в процессе работы программы, то ручной способ управления предпочтительнее.

При написании элементов управления, использующих мышь, важным является свойство ButtonDownDcn, которое имеется у всех графических объектов MATLAB. Щелкнув мышью на объекте figure или axes, можно изменить цвет фона, если выделен объект line, то можно изменить толщину и цвет линии. Для обработки щелчка мыши можно подготовить свои функции.

Утилита guide в MATLAB 6

В MATLAB 6 под Windows работа с guide в принципе не претерпела изменений, но сервисная часть значительно усовершенствована. При запуске команды guide появляется графическое окно с меню и набором иконок для установки графических объектов (левая часть рисунка) и редактирования интерфейса (правая сторона верхнего ряда), а также ряда стандартных для Windows значков. В этом окне происходит компоновка объектов. Представим простой пример подготовки графического интерфейса. На рис. 16.13 приведено окно, в котором введены три объекта: окно для вывода графиков (объект axes), список (объект listbox), кнопка (объект pushbutton).

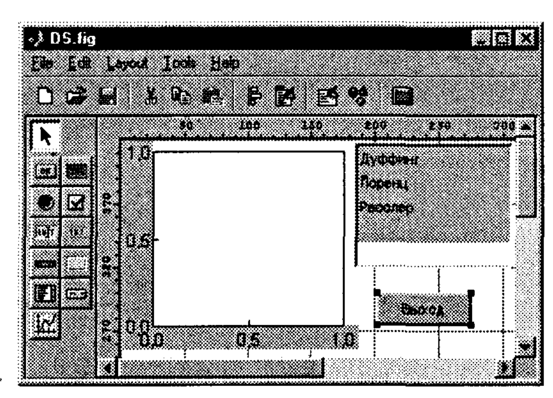


Рис. 16.13. Окно разработки графического интерфейса

Чтобы установить объект, выделим соответствующую ему иконку, а затем укажем местоположение объекта в рабочем окне. Расположение и размер легко интерактивно изменить, используя стандартные для Windows способы растяжения-сжатия и перемещения. Для просмотра иерархии установленных объектов полезно использовать специальный Object Browser (см. рис. 16.14), из которого легко перейти к редактированию свойств нужного объекта.

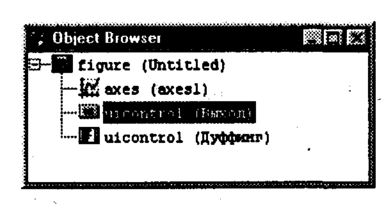


Рис. 16.14. Просмотр объектов

Окно Property Inspector предоставляет доступ к свойствам объектов создаваемого интерфейса. Изменения, производимые в этом окне, отображаются в окне guide. На рис. 16.15 представлено окно Property Inspector для интерфейса, изображенного на рис. 16.13. Для некоторых свойств достаточно выбрать из предлагаемого перечня, как, например, для свойства подсветки SelectionHightlight, при определении других – нужно задать числовые данные (Position). Для ряда свойств требуется ввести имя, например на рис. 16.15 это название кнопки для свойства String.

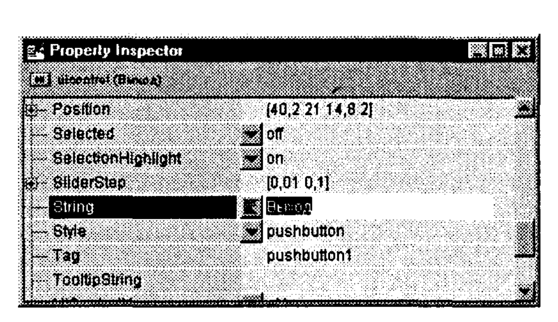


Рис. 16.15. Инспектор свойств

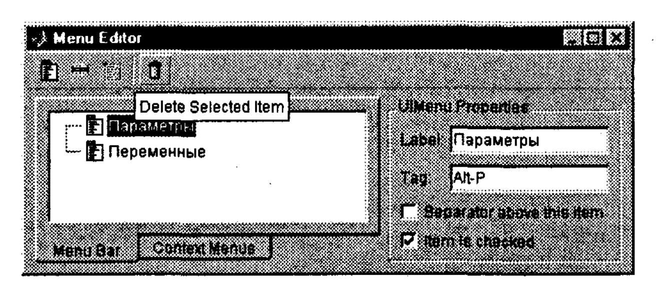


Рис. 16.16. Редактор меню

Кроме того, набор специальных окон поддерживает процесс создания интерфейса. Для подготовки меню имеется простой редактор Menu Editor, см. рис. 16.16.

Для задания расположения объектов в графическом окне можно использовать редактор выравнивания Align Objects, см. рис. 16.17.

Определению сетки и линеек помогает окно Grid and Rules, см. рис. 16.18.

Интерактивная разработка интерфейса при помощи guide состоит в подготовке макета, а процесс завершается созданием файла интерфейса (m-файл), в котором фиксируются сделанные интерактивно назначения. По завершении работы с программой появляется запрос на сохранение подготовленного файла. Сохраняются два файла: файл графического окна со всеми назначениями и специальный m-файл для вызова приложения (команда `Fig=DS`). Например, для представленного на рис. 16.12 интерфейса были записаны файлы DS.fig и DS.m. На рис. 16.19 приведено окно с фрагментом подготовленного файла DS.m. Этот файл можно вызывать и редактировать, как и всякий m-файл.

Чтобы продолжить редактирование интерфейса для рассмотренного примера в следующем сеансе, нужно запустить команду `guide DS`.

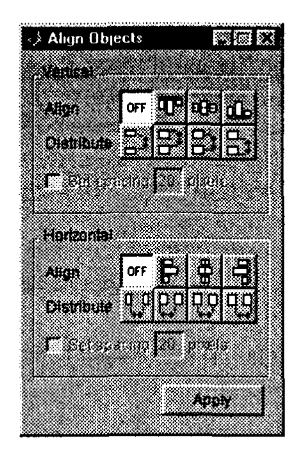


Рис. 16.17. Окно выравнивания

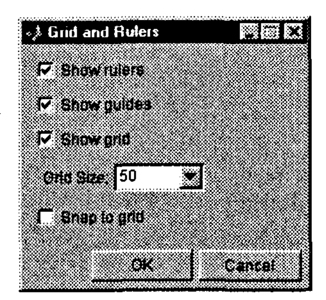


Рис. 16.18. Окно установки сетки и линеек

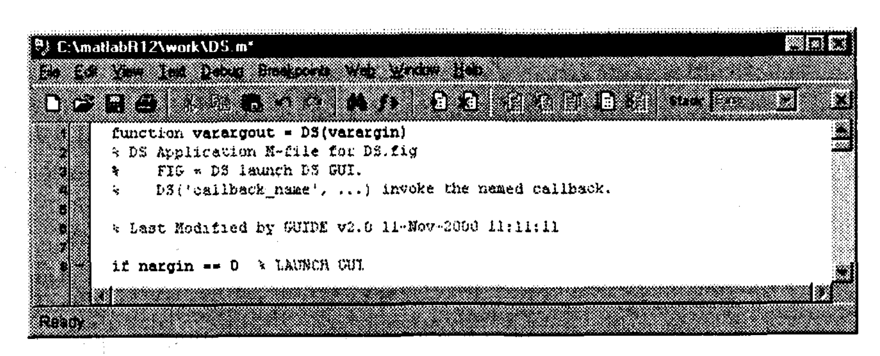


Рис. 16.19. Редактор с файлом интерфейса

Во всяком интерфейсе важную роль играют функции, которые должны вызываться при выборе пунктов меню, нажатии кнопок и т. д. Аппарат вызовов Callback приобрел новые возможности, и в частности, guide автоматически добавляет прототипы подфункций в файл интерфейса и устанавливает обращение к ним в свойстве Callback. Обращение к такой подфункции имеет следующий вид:

```
SUB_NAME(H, EVENTDATA, HANDLES, VARARGIN)
```

Здесь SUB_NAME — имя подфункции, которое составляется из двух частей: ярлыка объекта и типа вызова **Callback**, например "**lisbox1_Callback**" или "**figure1_CloseRequestFcn**". Дескриптор H определяется при помощи функции gcbo. Параметр EVENTDATA зарезервирован для последующего использования, а параметр HANDLES является структурой, содержащей дескрипторы интерфейса. Сама структура создается командой **guide** при помощи функций guihandles и guidata, а ее копия передается каждому вызову **Callback**. Саму структуру можно использовать для передачи дополнительной информации и менять при вызовах. Для этого следует запустить команду guidata(P.HANDLES) после изменения копии структуры. Для передачи дополнительных параметров предусмотрен аргумент VARARGIN.

По умолчанию MATLAB устанавливает в свойстве **Callback** следующий вызов:

```
DS("SUB_NAME", gcbo, [], guidata(gcbo))
```

17 ГЛАВА Расширения MATLAB

Пакеты MATLAB или инструментальные наборы Toolboxes являются уникальными собраниями процедур для решения самых разнообразных задач. Toolboxes в основном написаны на языке MATLAB, а их предметные области простираются от обработки сигналов (Signal Proccessing) и изображений (Image Processing) до проблем финасовой математики. Имеются пакеты для решения задач оптимизации Optimization, управления Controls, моделирования нейросетей Neural Net, подсистема компьютерной алгебры Symbolic Math, пакет статистики Statistics, пакет для решения уравнений в частных производных PDE и т. д. Все эти пакеты не входят в стандартный комплект и поставляются отдельно.

Многим пакетам посвящены специальные издания, и, естественно, невозможно в одной книге содержательно представить все пакеты. Основное место в этой главе отведено пакету Symbolic Math Toolbox — адаптированной к языку MATLAB версии системы Maple V фирмы Waterloo Maple Inc. Затем коротко описана работа с системой иследования динамических моделей SIMULINK, рассмотрено содержание пакета PDE и дан пример обработки изображения средствами пакета Image Processing, в конце главы приведен тематический обзор пакетов. Свежую информацию о расширениях MATLAB можно найти на сайте компании-производителя Mathworks Inc., см. главу 21 «Интернет и математика».

Пакет Symbolic Math

Возможности пакета Maple и многообразие реализуемых с его помощью операций достаточно подробно рассмотрены в первой части книги. В пакете Symbolic Math Toolbox представлено подмножество команд Maple, причем обращение к некоторым командам отличается от принятого в Maple. Поэтому в данном разделе мы в основном остановимся на особенностях работы с пакетом Symbolic Math, не перечисляя всех возможностей самих функций, родственных процедурам Maple. Для получения перечня доступных в MATLAB команд Maple можно обратиться к справочной системе или набрать в строке ввода команду

```
help symbolic
```

В процессе работы результаты выполнения команд Symbolic Math Toolbox присваиваются переменным MATLAB и могут затем использоваться в режиме вычислений, визуализации и т. д. Пакет замещает одноименные команды MATLAB, поэтому для получения справки о символьной команде FUNCTION из пакета Symbolic Math Toolbox следут предварить имя команды префиксом sym:

```
help sym/FUNCTION
```

Чтобы получить справку пакета Maple о команде FUNCTION, необходимо обратиться к справочной системе при помощи специальной команды

```
mhelp FUNCTION
```

Например, справка MATLAB о команде дифференцирования diff будет выведена по команде

```
help sym/diff
```

а справка Maple — по запросу

```
mhelp diff
```

Символьный объект

Для работы с командами ядра Maple в MATLAB определен новый тип sym — символьный объект (symbolic object). Фактически это строковые переменные. Для проведения аналитических операций интегрирования, дифференцирования и других нужно, чтобы соответствующие переменные были предварительно объявлены. Группа символьных переменных может быть создана по команде syms. Например, следующая команда опишет переменные s1 и s2:

```
» syms s1 s2
```

Другой способ определения символьного объекта предоставляет команда sym — аналог операции s3:='s3' в Maple:

```
» s3=sym("s3").
s3 =
s3
```

Эта команда позволяет также задать новую символьную переменную, связав ее с выражением:

```
» s4=sym(exp(s1)*s3+sqrt(3))
s4 =
exp(s1)*s3+3^(1/2)
```

В MATLAB отсутствует математическая нотация для представления формул, подобная той, что имеется в Maple. Для вывода аналитического выражения в математической форме служит функция pretty:

```
» pretty(s4)
                                           1/2
                              exp(s1) s3 + 3
```

Обнаружить символьные переменные в выражении позволяет команда findsym:

```
» findsym(s4)
```

```
ans =
s1, s3
```

Если символьной переменной S присвоено значение, то функция sym(S,'параметр') обеспечивает числовое преобразование в соответствии с указанным параметром. Вызов sym(S,'f') дает число в формате MATLAB, sym(S,'e') — число с оценкой отклонения от точного значения при помощи константы eps, sym(S,'d') — значение с числом знаков, сответствующим текущей величине параметра digits (32 знака по умолчанию). Например:

```
» s1=2^(1/2), sym(s1,'f'), sym(s1,'e'), sym(s1,'d')
s1 =
   1.4142
ans =
"1.6a09e667f3bcd"*2^(0)
ans =
sqrt(2)+64*eps/147
ans =
1.4142135623730951454746218587388
```

В Maple предусмотрена возможность проводить расчеты, используя арифметику произвольной точности. Для этого используются команда vpa и константа digits, задающая число значащих цифр. Например:

```
» vpa(pi,50)
ans =
3.1415926535897932384626433832795028841971693993751
```

При помощи описателя syms и команды sym реализуется также доопределение свойств символьной переменной. Опишем x и y как вещественные переменные:

```
» syms x y real
```

Определим комплексное число и вычислим его модуль:

```
» z=x +i*y; az=abs(z)
az =
(x^2+y^2)^(1/2)
```

Уберем информацию о вещественности переменной x и применим ту же операцию вычисления модуля. В результате получим:

```
» x=sym("x",'unreal'); abs(z)
ans =
abs(x+i*y)
```

Команда sym используется также для введения абстрактной функции:

```
» f=sym("f(x)")
f =
f(x)
```

Аналитические преобразования

Работа с выражениями MATLAB отличается от принятого в Maple стиля. Например, определим модуль комплексного выражения

```
» syms x y real, az=abs(x+i*y);
```

Присвоим переменной y числовое значение и обратимся к az, куда входит переменная y

```
» y=1; az
az =
(x^2+y^2)^(1/2)
```

Автоматической подстановки не произошло; в MATLAB нужно обратиться к команде subs, чтобы участвующим в выражении z переменным были присвоены те значения, которые они получили:

```
» subs(az)
ans =
(x^2+1)^(1/2)
```

Команда subs имеет следующий формат

```
subs(S,OLD,NEW)
```

Здесь S — выражение, в котором производится подстановка выражений NEW вместо OLD, где OLD и NEW — символьные переменные, строки (взятые в апострофы переменные или выражения) или массивы ячеек. Если массивы ячеек OLD и NEW одинакового размера, то каждый элемент OLD заменяется соответствующим элементом NEW. Также допускается обратный порядок следования параметров OLD и NEW.

Определим новую символьную переменную k и отменим назначение для y. После этого выполним операцию подстановки:

```
» syms y, k; az1=subs(az,[cos(k),sin(k)],[x,y])
az1 =
(cos(k)^2+sin(k)^2)^(1/2)
```

Автоматического упрощения также не произошло. Используем команду simplify:

```
» simplify(az1)
ans =
1
```

Кроме мощной многоцелевой команды simplify в MATLAB имеется интересная команда simple, дающая возможность посмотреть результаты применения различных упрощающих операций. В конце перечня возможных вариантов дается наилучший, по мнению системы, ответ.

Рассмотрим действие команды на элементарном примере возведения в квадрат переменной az1. Для экономии места отредактируем вывод результата MATLAB, поместив на строке с именем команды преобразования сам результат и убрав некоторые пробелы:

```
» simple(az1^2)
simplify: 1
radsimp: cos(k)^2+sin(k)^2
combine(trig): 1
factor: cos(k)^2+sin(k)^2
expand: cos(k)^2+sin(k)^2
combine: 1
```

```
convert(exp):
(1/2*exp(i*k)+1/2/exp(i*k))^2-1/4*(exp(i*k)-1/exp(i*k))^2
convert(sincos): cos(k)^2+sin(k)^2
convert(tan):
(1-tan(1/2*k)^2)^2/(1+tan(1/2*k)^2)^2+ 4*tan(1/2*k)^2/(1+tan(1/2*k)^2)^2
collect(k): cos(k)^2+sin(k)^2
ans =
1
```

В первой части, посвященной описанию системы Maple, дано описание команд, использованных при проведении упрощений. Аналитические операции для преобразования выражений перечислены в табл. 17.1.

Таблица 17.1. Аналитические операции с выражениями

Имя	Назначение
subs	Подстановка
subexpr	Запись с подстановками и использованием промежуточных выражений
simplify	Упрощение выражений
simple	Упрощение выражений с перечислением вариантов
expand	Раскрытие скобок
factor	Разложение выражения на множители
collect	Преобразование выражения в полином с выделением коэффициентов при степенях заданных переменных
numden	Приведение к рациональной форме
horner	Приведение к схеме Горнера

Команды анализа

Операции математического анализа Maple достаточно подробно описаны в первой части книги, а команды их вызова из MATLAB практически идентичны рассмотренным. Здесь мы ограничимся перечислением основных функций (табл. 17.2) и рядом примеров.

Таблица 17.2. Команды математического анализа

Имя	Назначение
diff	Вычисление производной
int	Вычисление интеграла
limit	Вычисление предела
symsum	Вычисление суммы ряда
taylor	Разложение выражения в ряд Тейлора
jacobian	Вычисление матрицы Якоби
finverse	Обращение функции
compose	Суперпозиция функций

Определим несколько символьных переменных и зададим выражение:

```
» syms x z k, f=k+sin(x)+besselj(2,z)
f =
sin(x)+besselj(2,z)
```

По умолчанию дифференцирование ведется по x:

```
» diff(f)
ans =
cos(x)
```

При дифференцировании по другим переменным их надо явно указывать:

```
» dz=diff(f,z)
dz =
besselj(1,z)-2/z*besselj(2,z)
```

Вторая производная вычисляется следующим образом:

```
» df2=diff(f,x,2)
df2 =
-sin(x)
```

Приведем пример получения якобиана для вектор-функции из двух элементов:

```
» jacobian([x*z; z^3],[z; x])
ans =
[      x,      z]
[ 3*z^2,      0]
```

При вычислении неопределенных и определенных интегралов обозначения естественны:

```
» int(x^k,x)
ans =
x^(k+1)/(k+1)
```

Обозначения бесконечности inf и числа π стандартны для MATLAB:

```
» int(exp(-k^2),k,0,inf)
ans =
1/2*pi^(1/2)
```

Для операции суммирования используется новое имя symsum, чтобы избежать конфликта с операцией sum системы MATLAB:

```
» symsum(k^(-2),1,inf)
ans =
1/6*pi^2
```

По умолчанию функция taylor вычисляет разложение в ряд Тейлора до пятой степени, а для указания иного порядка нужно определить третий параметр команды:

```
» [taylor(sin(x)),taylor(sin(x),x,9)]
ans =
[x-1/6*x^3+1/120*x^5, x-1/6*x^3+1/120*x^5-1/5040*x^7]
```

Заметим, что в этом примере и ряде последующих для экономии места результаты выполнения нескольких команд заключены в квадратные скобки для получения

вектора-строки. При вычислении предела может появиться нечисловая константа NaN:

```
» [limit(1/x,x,0), limit(1/x,x,0,'left')]
ans =
[ NaN, -inf]
```

Алгебра

Для алгебраических операций с символьными объектами используются команды (см. табл. 17.3) с именами, аналогичными тем, что применяются для числовых переменных (класс double). Однако, обнаружив, что аргументом той или иной команды выступает символьный объект, MATLAB запускает соответствующую процедуру пакета Symbolic Math Toolbox.

Таблица 17.3. Команды линейной алгебры

Имя	Назначение
diag	Задание или извлечение диагональных элементов матриц
triu	Выделение верхней треугольной матрицы
tril	Выделение нижней треугольной матрицы
inv	Обращение матрицы
det	Вычисление детерминанта матрицы
rank	Вычисление ранга матрицы
jordan	Вычисление канонической формы Жордана
rref	Приведение матрицы к верхней треугольной форме
null	Нуль-пространство матрицы
colspace	Базис-пространство столбцов
eig	Вычисление собственных значений и собственных векторов матриц
svd	Сингулярное разложение матриц
poly	Вычисление характеристического полинома матриц
expm	Вычисление матричного экспоненциала

Рассмотрим ряд простых примеров. Очистим текущие назначения сеанса и опишем переменные a, b, c как символьные:

```
» clear, syms a b c
```

Определим вектор:

```
» v = [a b]
v =
[ a, b]
```

Сформируем матрицу с главной диагональю из элементов этого вектора:

```
» diag(v)
ans =
[ a, 0]
[ 0, b]
```

При помощи дополнительного параметра команды diag подготовим матрицу с вектором, задающим наддиагональ (положительный второй параметр diag) и поддиагональ (отрицательное число):

```
» M=c*eye(3)+diag([a b],1)+diag([-b a],-1)
M =
[  c,  a,  0]
[ -b,  c,  b]
[  0,  a,  c]
```

Вычислим жорданову форму полученной матрицы:

```
» jordan(M)
ans =
[ c, 1, 0]
[ 0, c, 1]
[ 0, 0, c]
```

При помощи операций плавающей арифметики трудно вычислить каноническую жорданову форму, поскольку она чувствительна к возмущениям, но ее можно найти, используя аналитические преобразования. Вообще, наличие символьного анализатора позволяет проводить в MATLAB вычисления, переходя от операций с плавающей точкой (запятой) к аналитическим преобразованиям и обратно.

В качестве иллюстрации рассмотрим несколько операций с матрицей Гильберта второго порядка. Введем матрицу при помощи команды MATLAB:

```
» H2=hilb(2)
H2 =
    1.0000    0.5000
    0.5000    0.3333
```

Преобразуем вещественные числа в дроби, используя команду sym:

```
» H=sym(H2)
H =
[   1, 1/2]
[ 1/2, 1/3]
```

Найдем собственные значения. Поскольку элементами матрицы являются символьные объекты, то MATLAB обратится к команде из ядра Maple:

```
» ei=eig(H)
ei =
[ 2/3+1/6*13^(1/2)]
[ 2/3-1/6*13^(1/2)]
```

Вычислим спектр исходной матрицы:

```
» ei=eig(H2)
ei =
    1.2676
    0.0657
```

Для сравнения переведем результат аналитического вычисления спектра в десятичные числа:

```
» vpa(ei,5)
```

```
ans =
[    1.2676]
[ .65741e-1]
```

Отметим, что полученные числа являются символьными объектами. Если команда не предназначена для работы с символьными объектами, то будет выведено соответствующее сообщение. Например, для команды norm получим:

```
» norm(H)
??? Error using ==> norm
Function "norm" not defined for variables of class "sym".
```

Напомним, что подстановка значения элемента матрицы производится при помощи команды subs:

```
» M0=subs(M,c,0)
M0 =
[  0,  a,  0]
[ -b,  0,  b]
[  0,  a,  0]
```

Ядро полученной матрицы (нуль-пространство) вычисляется с помощью команды null, и при выводе результата для превращения вектора-столбца в строку используется транспонирование:

```
» null(M0)'
ans =
[-1, 0, 1]
```

Вычисление сингулярных чисел производится при помощи команды со стандартным именем, но работает процедура из Maple:

```
» svd(M0)
ans =
[                         0]
[ 2^(1/2)*(b*conj(b))^(1/2)]
[ 2^(1/2)*(a*conj(a))^(1/2)]
```

В ответе использована функция conj для обозначения сопряженной величины.

Решение уравнений

Для аналитического решения алгебраических уравнений используется команда solve, вызов которой имеет следующий вид:

```
solve(EQ1,EQ2,...,VAR1,VAR2,...)
```

Здесь EQ1, EQ2, ... — уравнения из символьных переменных или строки, задающие уравнения, VAR1, VAR2, ... — неизвестные (символьные переменные). При решении системы уравнений для хранения результата создается структура с полями, соответствующими именам неизвестных. Можно также использовать вариант вызова с неизвестными в качестве выходных параметров:

```
[VAR1,VAR2,...]=solve(EQ1,EQ2,...)
```

Приведем пример решения системы двух уравнений относительно неизвестных x и y.

```
» s=solve("x=y^3",'x+y=1')
s =
    x: [3x1 sym]
    y: [3x1 sym]
```

Выведем одно из решений для переменной x:

```
» s.x(1)
ans =
-1/6*(108+12*93^(1/2))^(1/3)+2/(108+12*93^(1/2))^(1/3)+1
```

Два других решения более громоздки. Если для вывода результата применить команду pretty(s.x), то все равно получатся довольно объемистые выражения, использующие обозначения повторяющихся частей (вспомогательные выражения %1, %2). В этом случае для преобразования результата вычисления корней уравнений полезна команда subexpr (см. табл. 17.1):

```
» [r,q]=subexpr(s.x,'q'); r', q
ans =
   0.3177              1.3412 + 1.1615i   1.3412 - 1.1615i
q =
[ 1/(108+12*93^(1/2))^(1/3)]
[   (108+12*93^(1/2))^(1/3)]
```

Переменная q дает вспомогательные выражения %1, %2 из аналитического выражения для решения, а выходной параметр r содержит сами корни. Действительно,

```
» double(s.x)'
ans =
   0.3177              1.3412 + 1.1615i   1.3412 - 1.1615i
```

Отметим, что команда solve в MATLAB не решает неравенств в отличие от команды, входящей в состав Maple.

Для решения дифференциальных уравнений в MATLAB применяется универсальная команда dsolve. Решение задач с начальными данными (задача Коши) и краевых задач при помощи команды dsolve проводится аналогично тому, как это описано в первой части книги, см. главу 4 «Решение уравнений в Maple». По умолчанию независимой переменной считается t. Для обозначения производных в уравнениях используется символ D, а комбинации D2, D3, ... применяются для обозначения второй, третьей и последующих производных.

Приведем примеры. Определим систему двух дифференциальных уравнений:

```
» de = "D2u=D1u-u+v, Dv=u"
de =
D2u=D1u-u+v, Dv=u
```

Решим задачу Коши для данного уравнения и двух начальных условий:

```
» S = dsolve(de, "u(0)=0, v(0)=0")
S =
    u: [1x1 sym]
    v: [1x1 sym]
```

Решение представлено структурой с двумя полями (по числу функций). Вычислим решение для функции u:

```
» S.u
ans =
1/2*C3*sin(t)-1/2*C3*cos(t)+1/2*C3*exp(t)
```

Поскольку для системы третьего порядка были заданы только два начальных условия, то одна константа осталась неопределенной — C3.

Теперь покажем, что команда dsolve позволяет находить решения краевой задачи. Выполним команду, поставив два условия в точке t=0 и одно в точке t=π:

```
» S = dsolve(eq,'Du(pi)=0, v(0)=0, u(0)=2'); [S.u,S.v]
ans =
[ 2*cos(t), 2*sin(t)]
```

Графика

Графика пакета Symbolic Math соответствует состоянию версии Maple V R5. Расширение MATLAB командами Maple привело к появлению новых графических функций, см. табл. 17.4.

Таблица 17.4. Графические команды Symbolic Math Toolbox

Имя	**Назначение**
ezplot	Построение графиков
funtool	Графический калькулятор функций одной переменной
ezpolar	График в полярных координатах
ezcontour	Рисование линий уровня
ezcontourf	Рисование линий уровня с заполнением
ezmesh	Построение сетчатой поверхности
ezmeshc	Построение поверхности с нанесением линий уровня
ezsurf	Построение поверхности
ezsurfc	Построение поверхности с нанесением линий уровня
ezplot3	Рисование трехмерных кривых

Команда ezplot применяется для рисования графиков функции одной переменной, неявно заданной функции двух переменных и параметрически заданной кривой. Для изображения графика функции f одной переменной на интервале [xmin xmax] в диапазоне изменения ординаты [ymin ymax] и в окне с номером fig используется следующий формат команды:

```
ezplot(f,[xmin xmax ymin ymax],fig)
```

Минимальный комплект для построения графика состоит из одного имени функции или аналитического выражения. При этом рисование производится в текущем окне, а если никакого окна не было открыто, то организуется графическое окно номер 1. Интервал независимой переменной по умолчанию есть [-2π,2π], а границы ymin и ymax выбираются пакетом автоматически.

Тот же формат имеет команда для построения неявно заданной функции f(x,y)=0. Чтобы построить график параметрически заданной кривой (x(t),y(t)), используется обращение

```
ezplot(x,y,[tmin tmax],fig)
```

Например, фигуру Лиссажу (рис. 17.1) можно получить при помощи следующего обращения к команде:

```
» ezplot("sin(3*x)",'cos(5*x)')
```

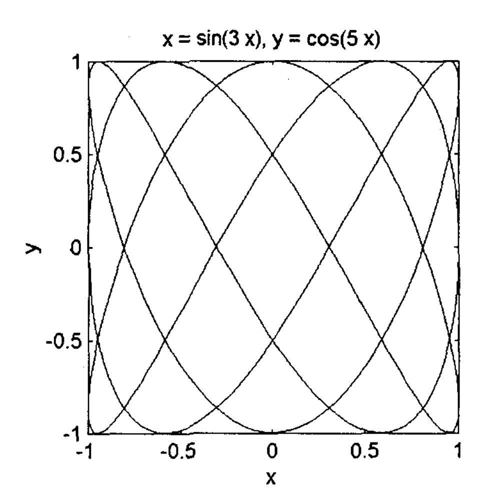


Рис. 17.1. Параметрически заданная кривая

Отметим, что маркировка осей и вывод заголовка производятся самой системой. Их можно убрать, используя инструменты графического окна или средства дескрипторной графики, см. главу 14 «Графика MATLAB». Дооформление графика осуществляется стандартными командами (`title`, `xlabel`, `legend`,...). Чтобы нарисовать несколько кривых, следует применять команду `hold` (сохранение рисунка для последующего вывода).

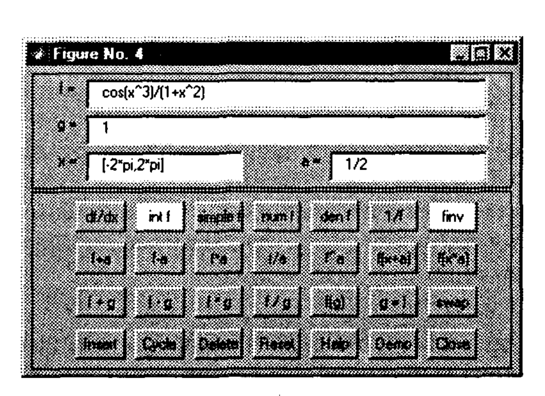


Рис. 17.2. Окно графического калькулятора funtool

Интересным инструментом для вывода графиков функций одной переменной является интерактивный графический калькулятор `funtool`, см. рис. 17.2. Введя в специальных окошках выражения для двух функций, можно получить их графики в отдельных окнах (на рис. 17.2 не приведены). Также просто сконструировать множество других функций, интегрируя, дифференцируя, вычисляя композицию

функций и используя еще несколько возможностей из указанных на панели funtool кнопок. Этот калькулятор предназначен для учебных целей, так же как и еще один калькулятор — команда taylortool для вычисления разложений в ряд Тейлора и изображения графиков исходной функции вместе с полиномом, представляющим отрезок ряда.

Разное

В заключение описания пакета Symbolic Math просто перечислим ряд команд, сгруппировав их по темам. В табл. 17.5 приведено несколько функций, расширяющих стандартный набор специальных математических функций.

Таблица 17.5. Специальные математические функции

Имя	Назначение
sinint	Интегральный синус
cosint	Интегральный косинус
rsums	Суммы Римана
zeta	Дзета-функция Римана
lambertw	Функция Ламберта (w-функция)

В табл. 17.6 представлены команды для проведения аналитических вычислений прямых и обратных трансформант для интегральных преобразований.

Таблица 17.6. Интегральные преобразования

Имя	Назначение
fourier	Прямое преобразование Фурье
ifourier	Обратное преобразование Фурье
laplace	Прямое преобразование Лапласа
ilaplace	Обратное преобразование Лапласа
ztrans	Z-преобразование
iztrans	Обратное Z-преобразование

В табл. 17.7 приведен ряд команд, позволяющих сохранить результаты аналитических преобразований в виде кода на языках программирования С и Фортран и текста для системы подготовки публикаций LaTeX. Здесь же представлены команды преобразования данных.

Таблица 17.7. Преобразование объектов

Имя	Назначение
latex	Запись выражений в стандарте LaTeX
ccode	Запись выражений на языке С

продолжение ⇨

Таблица 17.7 (*продолжение*)

Имя	Назначение
fortran	Запись выражений на языке Фортран
double	Преобразование символьной матрицы в числовую
poly2sym	Преобразование вектора коэффициентов полинома в символьный полином
sym2poly	Преобразование символьного полинома в вектор его коэффициентов
char	Преобразование символьного объекта в строковый

Наконец, несколько команд общего назначения дано в табл. 17.8.

Таблица 17.8. Работа с ядром Maple

Имя	Назначение
maple	Доступ к командам ядра системы Maple V
mfun	Численное вычисление Maple-функций
mfunlist	Вызов списка функций Maple V
mhelp	Справка по ядру Maple V
procread	Инсталляция Maple-процедур

Из MATLAB можно обращаться и к командам ядра Maple, которых нет в перечне функций Symbolic Math. Для этого служит команда maple. Например, для вычисления дискриминанта полинома можно использовать процедуру discrim, входящую в ядро Maple:

```
» maple("discrim",'x^3+1','x')
ans =
-27
```

Речь, однако, не идет о доступе к командам из библиотек системы Maple, даже если на компьютере установлены и Maple и MATLAB с пакетом Symbolic Math Toolbox.

SIMULINK

Среди расширений MATLAB особое место занимает SIMULINK — пакет для моделирования и анализа динамических систем. SIMULINK позволяет эффективно изучать разнообразные системы из технических, физических и иных приложений, рассматривать нелинейные задачи с непрерывным и дискретным временем. SIMULINK постоянно развивается, и в MATLAB 6 представлена уже четвертая версия. Среда SIMULINK обладает открытой архитектурой и интегрирована в MATLAB, что позволяет использовать графику и библиотеки MATLAB при исследовании моделей.

Графический интерфейс SIMULINK используется на всех стадиях моделирования и значительно облегчает работу. Этап подготовки модели заключается в сборке схемы процесса из готовых, заложенных в SIMULINK элементов и настройке параметров

системы. Чтобы упростить подготовку модели, применяются блочные диаграммы. Для введенной модели можно проводить собственно симуляцию, исследовать равновесия и выводить результаты в графическом и цифровом виде. Можно создавать новые элементы и библиотеки элементов, группировать их в блоки и создавать иерахические модели.

В SIMULINK имеется обширная библиотека элементов (блоков) для анализа линейных и нелинейных систем, дискретных, непрерывных и гибридных процессов, MIMO- и SISO-моделей. Допускается иерархическая структуризация моделей неограниченной вложенности. Для решения систем дифференциальных уравнений имеется несколько интеграторов, специальный акселератор для ускорения расчетов, можно также создавать и компилировать модули на С. Особенности имитационного моделирования при помощи пакета SIMULINK описаны в [11].

Блоковые диаграммы

Приведем простой пример моделирования динамики при помощи SIMULINK. Возьмем уравнение Дуффинга, которое демонстрирует возникновение стохастических колебаний для ряда нелинейных задач [65], в частности хаотическую динамику цепи с нелинейной индуктивностью, движение частицы в потенциале с двумя ямами или продольный изгиб упругой балки. В безразмерном виде уравнение (точка означает дифференцирование по времени t) записывается следующим образом:

$$\ddot{x} + kx + x^3 = B\cos t$$

Динамика определяется параметрами k и B, а также начальными условиями. В этой системе, варьируя параметры, можно наблюдать различные режимы движения: от периодических до хаотических.

Для моделирования необходимо подготовить систему. В командном окне можно вызвать SIMULINK, набрав `simulink` или щелкнув на соответствующем значке. В результате появится окно Simulink Library Browser, см. рис. 17.3. В этом окне представлены по группам элементы и блоки, из которых составляется модель. Здесь можно вызвать имеющуюся модель (файл с расширением mdl) или выбрать режим подготовки новой модели. Группы Continuous и Discrete служат соответственно для определения непрерывных и дискретных систем. Несколько групп позволяют задать функциональные зависимости и табличные данные (Functions & Tables), имеется набор математических функций (Math), различные нелинейные характеристики содержатся в группе Nonlinear, набор сигналов представлен группой Signals & Systems. Для указания выхода служит группа Sinks (раскрыта на рис. 17.3), а группа Sources предназначена для задания входных данных. (В MATLAB 6 окно несколько отличается от приведенного на рис. 17.3.)

Модель формируется в специальном окне, см. рис. 17.4, и работа происходит следующим образом: из окна Simulink Library Browser перетаскиваются нужные элементы и соединяются друг с другом. На рис. 17.4 представлена модель, отвечающая уравнению Дуффинга.

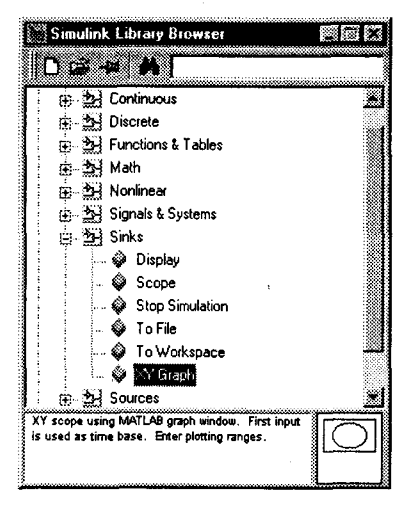


Рис. 17.3. Окно Simulink Library Browser

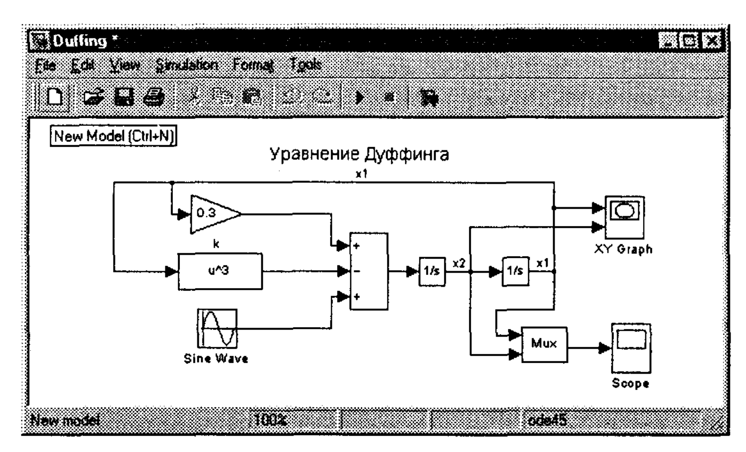


Рис. 17.4. Окно подготовки модели

Поясним назначение использованных на рисунке элементов. Заголовок «Уравнение Дуффинга» и надписи около соединительных линий и фигур являются текстовыми элементами и исполняют роль комментариев. Прямоугольник с плюсами и минусом представляет сумматор, трегольники определяют константы, прямоугольник с маркировкой u^3 представляет нелинейное слагаемое. Графики зависимости переменных от времени выводятся в специальное окно Scope, а фазовый портрет рисуется в окне XY Graph.

Для запуска симуляции можно использовать пункт Run меню Simulation или значок « ▶ ». В этом же меню предусмотрен пункт для определения параметров расчета. На рис. 17.5 приведено окно Scope с графиками зависимости переменной x и ее производной.

Видно, что полученный временной ряд не похож на периодичесие колебания. Изображение траектории на фазовой плоскости (элемент XY Graph) приводится на рис. 17.6.

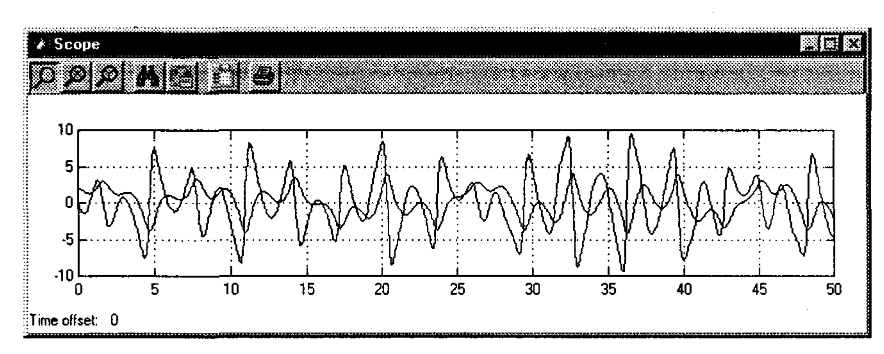


Рис. 17.5. Окно Scope

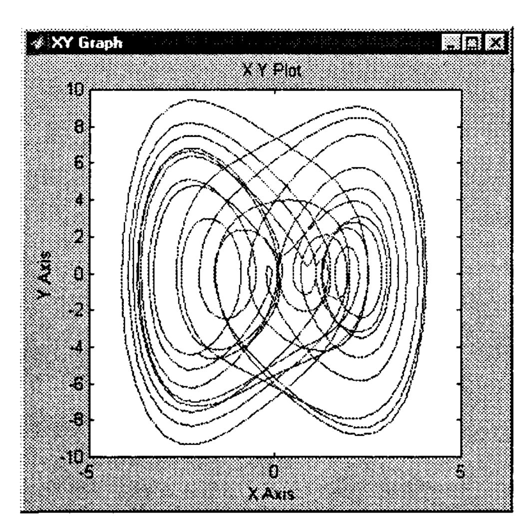


Рис. 17.6. Двумерное окно MATLAB для вывода траектории

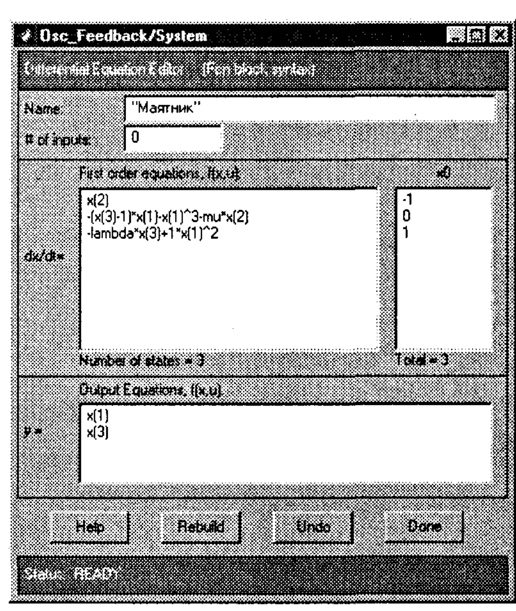
Редактор динамических систем

Имеются и другие способы задания моделей, требующие минимального использования блоков. Для системы обыкновенных дифференциальных уравнений можно подготовить модель при помощи специального редактора динамических систем DEE. В этом случае диаграмма состоит из специального блока DEE, в котором содержится описание модели в виде системы дифференциальных уравнений первого порядка и начальных данных, и блока вывода. Полученная таким образом система будет вычисляться с той скоростью, с которой работают встроенные блоки SIMULINK.

Рассмотрим работу с редактором DEE на примере подготовки модели, описывающей движение шарика с потенциалом, меняющимся из-за наличия обратной связи (см. главу 10 «Примеры решения задач»). Заполнение областей ввода представлено на рис. 17.7. В строке Name можно определить имя, которым будет помечен блок DEE на блоковой диаграмме, см. рис. 17.8. Число входных данных задается в стро-

ке # of inputs:, отрицательное число, введенное в эту строку, преобразуется в нуль. Правые части уравнений вводятся построчно в окне First order equations. Для обозначения вектора переменных используются компоненты векторов x, а u обозначает входной вектор данных (например, время, если нужна неавтономная система). При формулировании правой части системы могут применяться любые переменные из рабочей области MATLAB, любые функции, поддерживаемые в блоках Simulink Fcn block (тригонометрические, гиперболические, их обратные и др., полный список функций содержится в руководстве по SIMULINK). Область начальных данных обозначена X0:, а список выводимых параметров дается в окне Output Equations:. Параметры рассматриваемой системы дифференциальных уравнений (mu и lambda) достаточно ввести в командном окне MATLAB:

```
» mu=.3; lambda=.1;
```



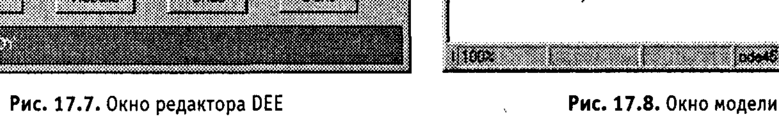


Рис. 17.7. Окно редактора DEE

Рис. 17.8. Окно модели

Кнопка Rebuild предназначена для реконструкции файла модели (расширение mdl) системы после коррекций, а назначение остальных кнопок (Help, Cancel, Done) стандартно.

Диаграмма модели (рис. 17.8) состоит из двух элементов: в блоке «Маятник» содержится собственно система дифференциальных уравнений, а для вывода результатов на фазовую плоскость добавлен элемент из библиотеки Simulink Library Browser (XY Graph). На рис. 17.9 приведена траектория, полученная в результате запуска с начальными данными после длительного установления. Здесь использована возможность получения расчетных данных в командном окне MATLAB. Для этого в окне параметров запуска (пункт Parameters... меню Simulation, см. рис. 17.10) был отмечен соответствующий пункт. Чтобы нарисовать траекторию, приведенную на рис. 17.9, было сделано несколько расчетов. Вначале использовались данные из

окна редактора. Затем было указано, что финальная точка доступна в командном окне, переменной xInitial присвоено содержимое вектора xFinal:

```
» xInitial=xFinal
xInitial =
  3.6356e-001 -1.0221e-001  1.0802e+000
```

Наконец, в окне параметров запуска был выбран пункт, указывающий в качестве начальных данных использовать xInitial. Результат запуска с этими данными представлен траекторией (рис. 17.9).

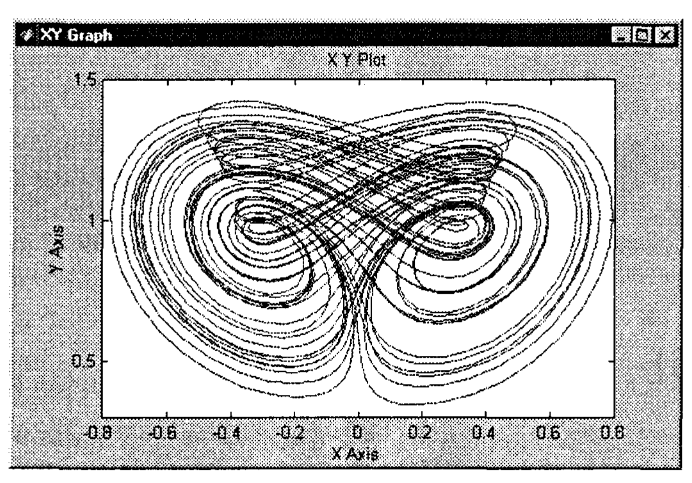


Рис. 17.9. Хаотическая траектория

Simulation Parameters: Osc_Feedback
Solver | Workspace I/O | Diagnostics
Load from workspace
Input: [t, u]
Initial state: xInitial
Save to workspace
Time: tout
States: xout
Output: yout
Final state: xFinal
Save options
Limit rows to last: 1000
Decimation: 1
Format: Matrix
OK | Cancel | Help | Apply

Рис. 17.10. Окно параметров модели

С помощью акселлератора SIMULINK Accelerator прямо из блоковой модели можно сгенерировать код на языке С и откомпилировать его. В результате будет получена версия модели, расчет которой потребует в несколько раз меньшего времени.

Это существенно, например, при использовании метода Монте-Карло, когда многократно, с разными начальными данными производится расчет динамики одной модели. К тому же код создается в стандарте ANSI C и является переносимым. Ускоритель поддерживает все стандартные интеграторы SIMULINK, отслеживает изменения структуры модели и запускает перекомпиляцию, позволяет изменять параметры модели без перекомпиляции. Генерируемый код оптимизируется, чтобы стало возможно моделирование в реальном масштабе времени. Это достигается оптимизацией циклов и вызовов функций, удалением единиц и нулей из арифметических выражений.

Из SIMULINK могут использоваться следующие пакеты: Real-Time Workshop, Stateflow, Stateflow Coder, DSP Blockset, Fixed-Point Blockset, Nonlinear Control Design Blockset. Например, с помощью Real-Time Workshop можно сгенерировать код на C и создавать программы, запускаемые автономно.

Система SIMULINK помимо графического интерфейса позволяет проводить эксперименты, используя командный режим (функция sim). Модели можно отлаживать при помощи функции sldebug, для задания параметров применяются функции simset и simget, использование которых аналогично работе с функциями set и get для определения параметров интегрирования обыкновенных дифференциальных уравнений.

Пакет PDE

Рассмотрим организацию пакета PDE Toolbox, предназначенного для решения уравнений в частных производных. Пакет составляют команды задания области и ее дискретизации, формирования и решения систем алгебраических уравнений, а также сервисные функции. Ряд процедур ориентирован на решение классов задач: гиперболических (hyperbolic), параболических (parabolic) и нелинейных уравнений (), определения спектра (pdeeig). Для решения уравнения Пуассона на прямоугольной сетке имеется функция poisolv.

Пакетом поддерживается дискретизация на треугольной неравномерной сетке, и решение задачи основывается на алгоритмах метода конечных элементов. Пакет включает функции построения треугольной сетки (initmesh), ее уточнения (refinemesh) и другие функции. Адаптивная сетка генерируется функцией adaptmesh, а для ассемблирования (формирования системы алгебраических уравнений) применяются процедуры assempde, assemb, assema. Специальные функции реализованы для часто используемых областей, например построение регулярной сетки для прямоугольника (poimesh). Имеются функции для записи спецификаций краевых условий (wbound) и параметров дискретизации (wgeom).

Для задания области применяются функции, определяющие окружность (pdecirc), эллипс (pdeellip), прямоугольник (pderect) и многоугольник (pdepoly). Кроме того, имеется специальное средство для построения интерфейса (pdetool). Набор графических функций позволяет дать изображение области (pdegplot), сетки (pdemesh), вывести результат в виде поверхности (pdesurf) и линий уровня (pdecont).

Вычислительные алгоритмы включают прямое и обратное дискретные синус-преобразования — dst и idst соответственно, а также быстрое решение уравнения Пу-

ассона на прямоугольной сетке (poicalc). Для вычисления обобщенной задачи на собственные значения для разреженной матрицы используется функция sptarn. Имеются функции для вычисления потоков, градиента, интерполяции функции, оценки погрешности, для интерполяции данных с треугольной сетки на прямоугольную (tri2grid). Последняя команда полезна при построении линий уровня и поверхностей стандартными графическими средствами MATLAB.

При решении двумерных задач удобно использовать команду общего назначения (pdeplot), которая обеспечивает графический интерфейс для всех этапов работы: выбора модели, задания области, определения граничных и начальных условий, решения и вывода результата. После запуска команды появляется окно, см. рис. 17.11, на котором показано построение сетки для области, полученной объединением двух эллипсов.

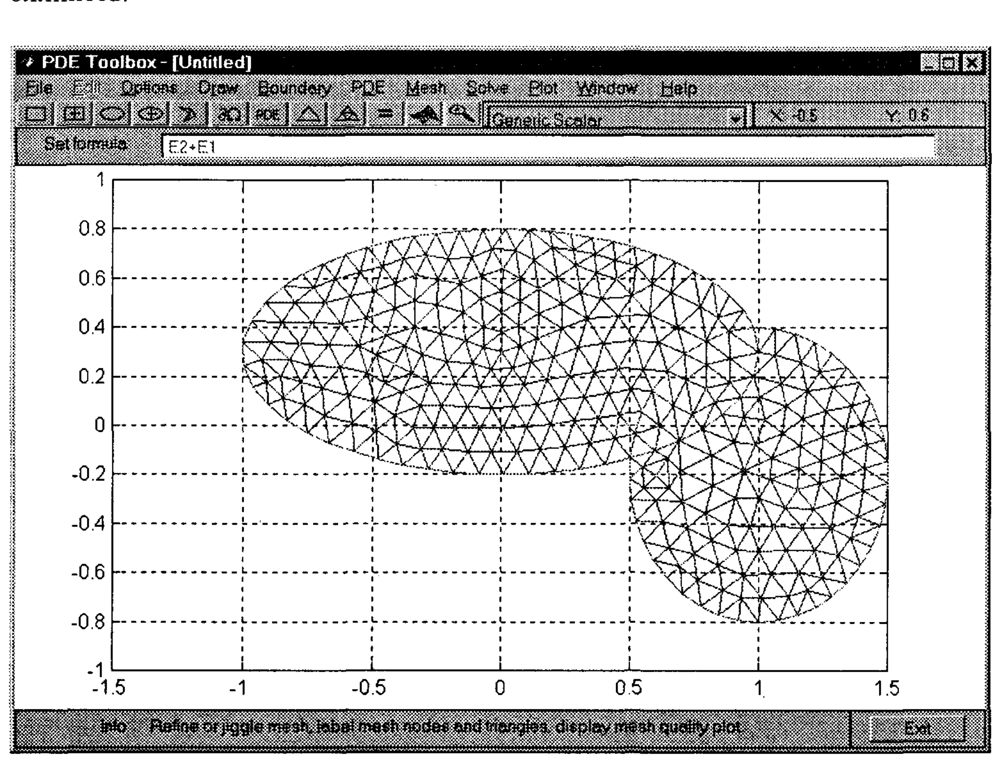


Рис. 17.11. Окно команды pdetool

Работать с pdeplot довольно просто. Пункт **Applications** (меню **Options**) позволяет выбрать тип задачи из списка: скалярное уравнение или система, уравнения теории упругости (плоское напряженное состояние или плоская деформация), уравнения электро- или магнитостатики, задача диффузии и др. Затем необходимо задать область (меню **Draw**). Даже геометрически сложная область может быть быстро подготовлена при помощи графического интерфейса и мыши. После этого следует задать краевые условия (меню **Boundary** или соответствующий значок). Здесь можно выделить часть границы и двойным щелчком мыши вызвать окно, в котором

указать тип (условие Дирихле или Неймана) и задать числовые значения. Далее следует конкретизировать вид уравнений (меню PDE или одноименный значок). Двойным щелчком мыши на каждой подобласти можно вызвать специальное окно (PDE Specification), см. рис. 17.12, в котором следует определить уравнение и назначить числовые коэффициенты.

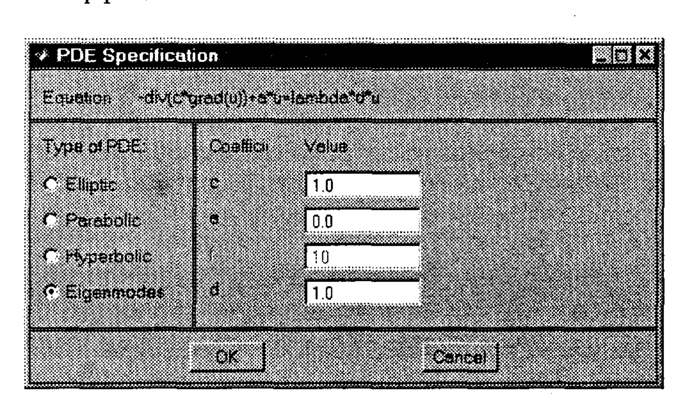


Рис. 17.12. Окно для задания уравнений (команда pdetool)

После этого производится дискретизация области треугольниками, причем можно уточнять сетку, оценивая ее качество графически и численно. Для определения расчетных параметров и запуска решения служит пункт меню Solve и соответствующий ему значок («равно»).

Приведенную последовательность действий можно менять, проходя этапы подготовки задачи в ином порядке. Например, сразу после задания области можно построить сетку, а затем перейти к определению уравнений и краевых условий. При работе pdetool вызывает функции пакета PDE для задания области, триангуляции, постановки краевых условий и т. д.

Результаты решения могут быть представлены графически в окне pdeplot или отдельных окнах MATLAB. На рис. 17.13 дан контурный график собственной функции для рассматриваемой задачи.

Подготовленные интерактивно в pdetool данные можно сохранить в виде m-файла модели (function pdemodel), куда запишутся геометрия области, краевые условия, сведения о сетке, коэффициенты уравнений и другие параметры задачи. Это позволит впоследствии продолжить работу с моделью.

Рассмотрим задачу о диффузии в квадратной области начального температурного пятна в форме двух холмов (восьмерка в плане). Будем решать уравнение теплопроводности, считая, что на границе заданы нулевые краевые условия (задача Дирихле):

```
» geom='squareg'; boun='squareb1';
```

По входной информации о геометрии рассматриваемой задачи функция initmesh проводит триангуляцию области и формирует три массива: вершин, сторон и треугольников:

```
» [poi,edg,tri]=initmesh(geom);
```

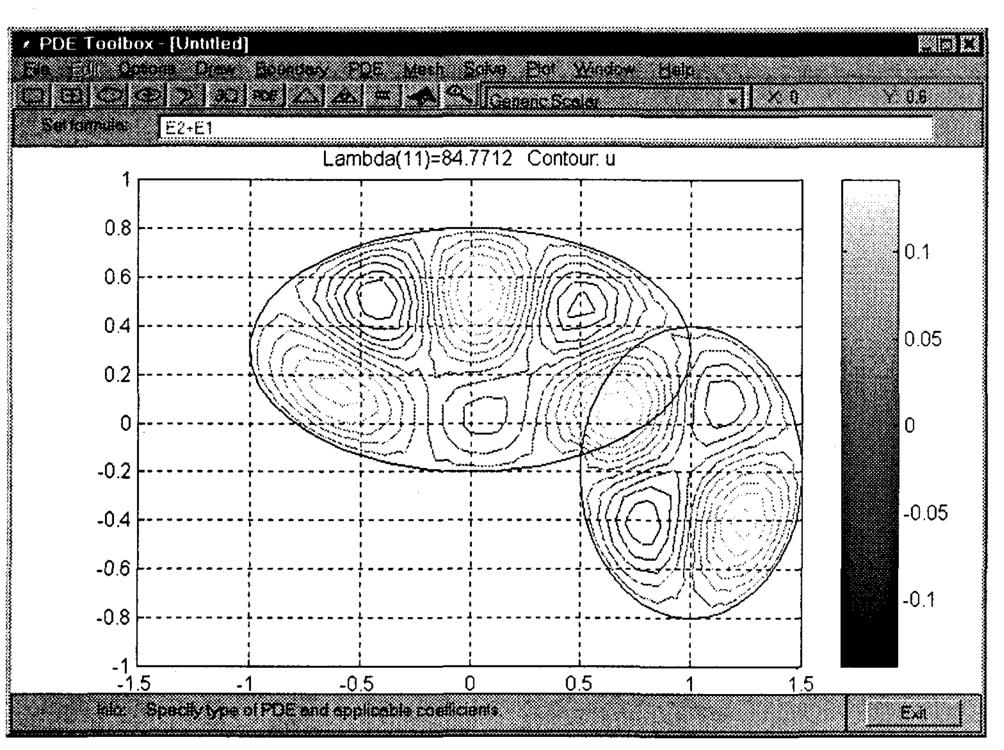


Рис. 17.13. Результат решения в окне pdetool

Введем начальную температуру:

```
» u0=exp(poi(1,:).^2-4*poi(1,:).^4-4*poi(2,:).^2-.4)';
u0=abs(sin(pi*poi(1,:)).*cos(pi/2*poi(2,:)))';
ix=find(poi(1,:).^2+poi(2,:).^2>0.8^2);
u0(ix)=zeros(size(ix));
```

Определим моменты времени для записи температуры (`tlist`) и обратимся к функции интегрирования параболической задачи `parabolic`:

```
» tlist=0:.05:1; x=linspace(-1,1,21); y=x;
u1=parabolic(u0,tlist,boun,poi,edg,tri,1,0,1,1);
```

В результате работы функции выводится информация о прохождении определенных списком `tlist` моментов времени, количестве выполненных шагов и т. д. Для нашего расчета было выведено:

```
93 successful steps
1 failed attempts
190 function evaluations
1 partial derivatives
21 LU decompositions
189 solutions of linear systems
```

Для изображения распределения температуры в отдельные моменты времени организуем цикл. Чтобы воспользоваться функциями `surf` и `contour`, полученные дан-

ные нужно перевести на прямоугольную сетку. Заметим, что при первом преобразовании (функция tri2grid) выводятся массив значений u и три дополнительных массива tn, a2, a3, которые используются при последующих обращениях:

```
» kk=0; k11=[1 2 6 size(u1,2)];
k0=length(k11); colormap(gray)
for k=k11, kk=kk+1;
    if kk==1, [u,tn,a2,a3]=tri2grid(poi,tri,u1(:,k),x,y);
    else u=tri2grid(poi,tri,u1(:,k),tn,a2,a3);
    end
    i=find(isnan(u)); u(i)=zeros(size(i));
    subplot(2,k0,kk), mesh(x,y,u,'EdgeColor',[0 0 0]);
    axis off equal, axis(.95*[-1 1 -1 1 0 1])
    subplot(2,k0,kk+k0),
    contour(x,y,u); axis off equal tight,
    title(strcat('t=',num2str(tlist(k))))
end
```

На рис. 17.14 представлены распределения температуры в некоторые моменты времени (верхний ряд) и линии уровня (нижний ряд). Видно, что информация о начальном распределении теряется очень быстро.

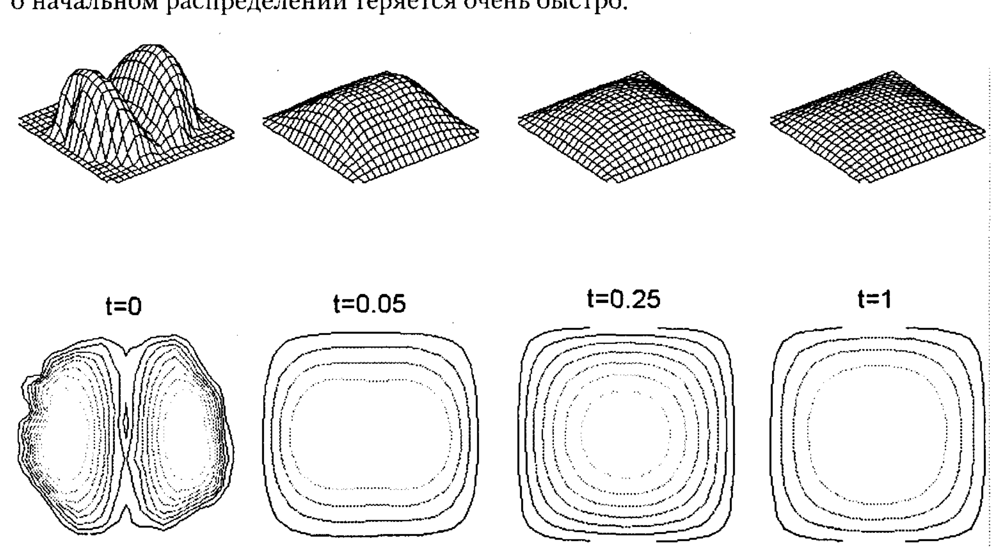


Рис. 17.14. Диффузия температурного пятна

Обработка изображений

Пакет Image Processing избавляет от кодирования при операциях анализа и преобразования изображений, восстановления и выделения деталей изображения, поскольку предоставляет унифицированную оболочку. Мы не будем перечислять имеющиеся команды, а ограничимся одним примером применения ряда функций, сопроводив их использование короткими пояснениями.

Приведем пример обработки изображения и выделения детали из рисунка, выбрав для этого картину С. Дали «Рынок рабов с невидимым бюстом Вольтера». Прочитаем фрагмент картины из файла в формате tiff и трансформируем rgb-палитру в индексированное изображение (команда rgb2ind), а затем удалим информацию о цвете, выбрав палитру из оттенков серого цвета (команда ind2gray):

```
[Pic,m]=imread("dalism.tif",'tiff');
[X,map]=rgb2ind(Pic,256); X=ind2gray(X,map); ss=size(X);
```

Выведем полученное изображение, см. рис. 17.15.

```
imshow(X)
```

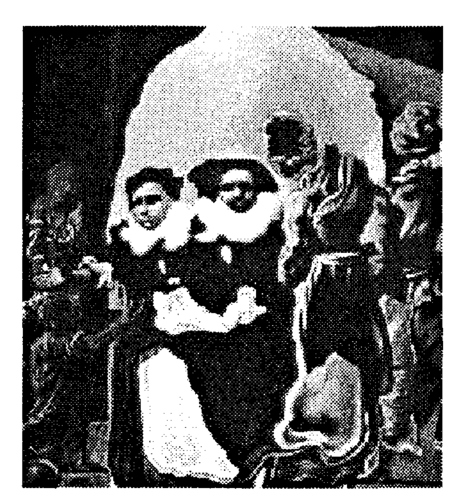


Рис. 17.15. Фрагмент картины

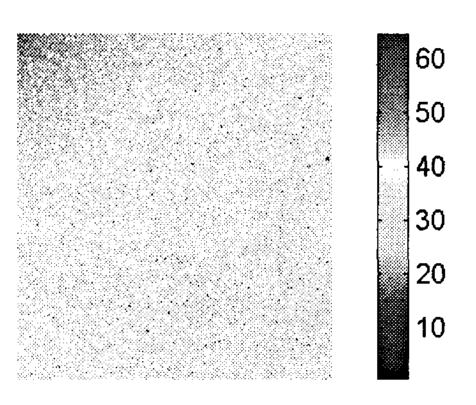


Рис. 17.16. Образ двумерного косинус-преобразования

Применим двумерное дискретное косинус-преобразование, вычислим логарифм от амплитуды и полученную матрицу превратим в файл изображения (команда mat2gray):

```
D=dct2(X); Dm=mat2gray(log(abs(D)));
```

Полученное таким образом изображение можно посмотреть, преобразовав предварительно массив Dm так, чтобы изображение было с индексированными цветами и палитрой jet, см. рис. 17.16:

```
figure, imshow(gray2ind(Dm,64),jet(64)); colorbar
```

Выделим наиболее существенную часть матрицы D и выполним для нее обратное косинус-преобразование. Нарисуем получившееееся, см. рис. 17.17:

```
DD=D(1:70,1:60);Xnew=idct2(DD,ss);
figure, imshow(Xnew)
```

Обработаем полученное изображение, используя усреднение по блокам 3×3 пиксела:

```
B=blkproc(Xnew,[3 3],'mean2(x)*ones(3,3)');
figure, imshow(B)
```

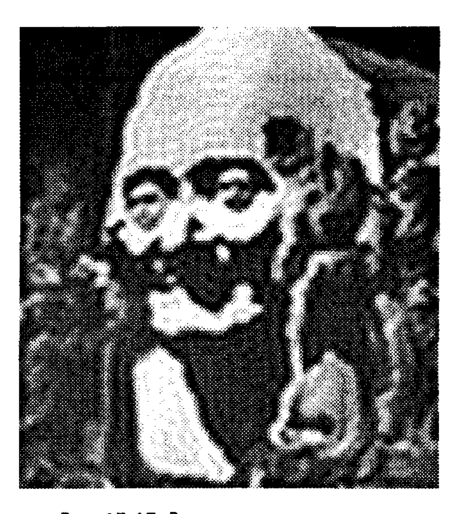


Рис. 17.17. Результат для уменьшенного набора частот

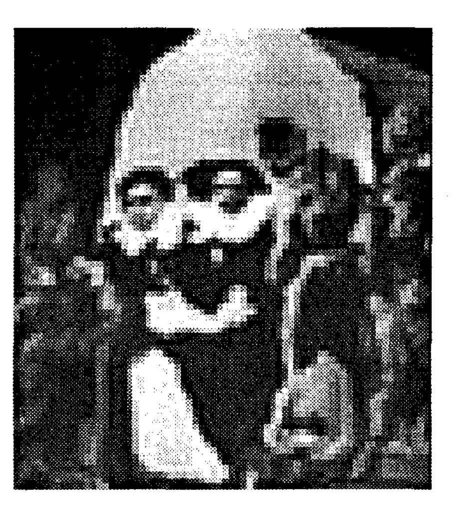


Рис. 17.18. Результат — бюст Вольтера

В результате получим изображение, на котором отчетливо различим бюст Вольтера, см. рис. 17.18.

Обзор пакетов

Для остальных пакетов MATLAB мы ограничимся краткими пояснениями, распределив их на три группы. Конечно, это условная классификация, поскольку многие функции, отнесенные к классу математических, могут успешно применяться для решения задач, относящихся к разным предметным областям. Некоторые пакеты разработаны достаточно давно и сегодня используются уже реализации под номерами от 2 до 5, а другие пакеты появились недавно.

Важно также то, что многие пакеты не только обеспечивают квалифицированную среду для разработки технических приложений и проведения научных исследований, но и предоставляют прекрасные возможности для обучения студентов различных специальностей. Качественные демонстрации, разработанные для пакетов MATLAB, служат дополнительным материалом по различным разделам научного знания.

Математические пакеты

Работа со сплайнами представлена набором Spline Toolbox. Кубические сплайны, кривые Безье, функции сглаживания, специальные процедуры для решения возникающих систем линейных алгебраических уравнений и другие функции составляют содержание этого пакета. Разнообразные демонстрации, дающие ясное представление об имеющихся функциях, могут быть вызваны командой spdemos.

Пакет Statistics обеспечивает широкий спектр инструментов для статистических вычислений, к тому же возможности пакета легко совершенствовать, создавая собственные методы статистики и анализа при помощи функций на языке MATLAB.

Пакет Optimization содержит программы минимизации линейных и нелинейных функций. Пакетом поддерживаются безусловная оптимизация нелинейных функций, метод наименьших квадратов и нелинейная интерполяция, решение нелинейных уравнений, решение задач линейного и квадратичного программирования, метод минимакса и многокритериальная оптимизация.

Для работы с вэйвлетами (wavelet) разработан пакет Wavelet Toolbox, который позволяет проводить множество операций. Команда waveinfo предоставляет информацию о доступных вэйвлетах, а для создания новых вэйвлетов имеется специальный менеджер — команда wavemngr. Одномерные и двумерные операции можно проводить, используя команды пакета и специальное меню, вызываемое по команде wavemenu. Для представления возможностей пакета вэйвлетом подготовлена демонстрация, запускаемая по команде wavedemo.

Специальные алгоритмы для анализа сигналов на основе моментов высокого порядка представлены в пакете Higher-Order Spectral Analysis. Функции пакета позволяют анализировать сигналы, поврежденные негауссовым шумом, и процессы, происходящие в нелинейных системах, обеспечивают восстановление гармоник, поддерживают линейные модели прогноза и оценку запаздывания.

Пакет LMI Control ориентирован на задачи линейного программирования, снабжен специальным графическим редактором и является мощным средством для решения выпуклых задач оптимизации. Имеет набор функциий для многокритериального проектирования регуляторов и анализа устойчивости динамических систем.

Для работы с нечеткими моделями предназначен пакет Fuzzy Logic, обеспечивающий поддержку методов нечеткой кластеризации и адаптивные нейронные сети. Пакет содержит несколько графических редакторов для представления информации в процессе создания нечетких моделей и предназначен для совместной работы с SIMULINK. С помощью пакета Real-Time Workshop можно сгенерировать код на С для ускорения обработки данных.

Помимо собственных средств обработки данных в MATLAB можно вызывать ресурсы электронной таблицы Excel фирмы Microsoft. Пакет Excel Link связывает Excel с рабочей областью MATLAB и обеспечивает следующие режимы взаимодействия двух систем: двусторонний обмен данными, просмотр, редактирование и обработка данных MATLAB в Excel и, наоборот, вызов функций MATLAB из Excel, подготовка Excel-приложений. В Excel удобно редактировать данные, а их обработку и визуализацию проводить в MATLAB, используя все семейство продуктов. Взаимодействие не требует низкоуровневого программирования и создания промежуточных файлов.

В пакете NAG Foundation собрано более двухсот функций библиотеки фирмы Numerical Algorithms Group Ltd. (NAG), предназначенных для решения задач линейной алгебры, статистики и оптимизации, аппроксимации и интерполяции, решения обыкновенных дифференциальных уравнений и уравнений в частных производных и т. д. Пакет содержит процедуры библиотеки NAG в виде кодов и соответствующих m-файлов для их вызова. Средство NAG Gateway Generator позволяет создавать программы на Фортране для подключения посредством динамических библиотек к MATLAB.

Инженерные пакеты

Многочисленные инструменты для обработки изображений собраны в пакете Image Processing Toolbox. О возможностях пакета дает представление одно перечисление групп, в которые организованы команды пакета: геометрические операции и анализ изображений, преобразования типов и цветов, одномерная и двумерная фильтрация, сжатие изображений, работа с пикселами и бинарные операции, статистика. Несколько прекрасных демонстраций вводят в многообразие средств обработки изображений, составляющих наполнение пакета Image Processing. Выше был дан пример использования пакета для выделения изображения, спрятанного в картине.

Пакет Signal Processing предназначен для обработки сигналов, анализа временных рядов и разработки соответствующих систем. Основные области применения пакета относятся к моделированию сигналов и линейных систем, проектированию цифровых и аналоговых фильтров, оценке спектров и статистической обработке сигналов. Пакет включает функции для реализации различных преобразований, используемых для анализа и фильтрации данных, а также кодирования. Это идеальный инструмент, в котором собраны максимально эффективные и надежные алгоритмы, проверенные многолетней практикой. Графическая оболочка пакета помогает анализировать сигналы, проводить спектральный анализ, проектировать фильтры. На основе пакета Signal Processing можно решать разнообразные задачи, привлекая средства других пакетов. Для обработки двумерных сигналов и изображений полезно участие пакета Image Processing. Для выделения классификационных характеристик возможно совместное использование средств пакетов Neural Network и Fuzzy Logic, а для параметрического моделирования во временной области понадобится пакет System Identification.

Для моделирования цифровых и аналоговых устройств (системы связи и передачи информации) предназначен пакет Communications Toolbox. Современные средства разработки, анализа и тестирования моделей поддерживаются набором более чем 100 функций MATLAB и 150 блоков SIMULINK. Использование пакета ускоряет проектирование, анализ и моделирование коммуникационных систем. Например, включенная в пакет модель модема v.34 полностью документирована и может быть использована для обучения студентов. Для создания прототипов плат цифровой обработки можно использовать Real Time Workshop.

Проектирование и моделирование непрерывных и дискретных систем автоматического управления поддерживаются в MATLAB набором функций пакета Control System Toolbox. Пакет состоит из функций, реализующих традиционные методы анализа передаточных функций и современные алгоритмы для работы в пространствах состояний. Пакет позволяет анализировать системы с непрерывным и дискретным временем, строить линейные модели систем. Он часто используется в комбинации с другими пакетами MATLAB.

Пакет Model Predictive Control предоставляет средства для реализации управления сложными многоканальными процессами при наличии ограничений на переменные состояния и возможности управления. Использован предиктивный подход, заключающийся в построении явной линейной динамической модели объекта для прогнозирования воздействий управления на поведение объекта. На каждом этапе решается оптимизационная задача квадратичного программирования с ограни-

чениями. Включены функции для обеспечения взаимодействия с пакетом System Identification и SIMULINK.

Для проектирования устойчивых систем управления разработан пакет μ-Analysis and Synthesis Toolbox. Пакет позволяет проектировать оптимальные в равномерной и интегральной норме регуляторы; имеются средства понижения порядка модели для упрощения операций с блоками, а также графический интерфейс.

Специализированные функции для идентификации динамических систем по временному или частотному сигналу собраны в пакет Frequency Domain Sytem Identification Toolbox. Пакет позволяет осуществлять диагностику моделей (моделирование и вычисление невязок), проводить идентификацию непрерывных и дискретных систем с неизвестным запаздыванием. Поддерживается преобразование моделей в формат пакета System Identification Toolbox и обратно.

Пакет Neural Network позволяет применять технологию нейронных сетей к задачам обработки сигналов, нелинейного управления и финансового моделирования. Пакет имеет модульную организацию, дает возможность использовать более 15 типов сетей и обучающих правил, снабжен функциями инициализации для каждого типа архитектуры, а также демонстрациями.

Пакет Fuzzy Logic позволяет проектировать и диагностировать нечеткие модели, основанные на адаптивных нейронных сетях и методах кластеризации. Для интерактивного слежения за поведением систем имеются продуманный интерфейс и набор графических функций. Пакет предназначен для совместной работы с SIMULINK.

Для отображения географической информации разработан пакет Mapping Toolbox. Возможности пакета включают визуализацию, обработку и анализ графических данных, банк из более 60 картографических проекций, проектирование векторных, матричных и составных карт, трехмерное представление информации. Имеется удобный графический интерфейс, конверторы форматов географических данных, сопряжение с существующими базами данных и атласами. Имеется подсистема для решения геостатических и навигационных задач.

Финансовая математика

В последнее время все большее количество пользователей используют MATLAB как средство для эффективных экономических расчетов. Это связано с тем, что в MATLAB включены пакеты Financial, Statistics, Neural Network, Financial Time Series, GARCH, Optimization, имеющие финансово-экономические приложения.

Пакет Financial пригоден для решения множества финансовых задач и может быть использован для расчета процентных ставок и прибыли, анализа производных доходов и депозитов, оптимизации портфеля инвестиций. Например, средства пакета Financial позволяют строить диаграммы зависимости риска и рентабельности для портфеля инвестиций и предлагать распределение инвестиций, обеспечивающих планируемый доход с заданным уровнем риска.

Пакет Financial Time Series содержит инструменты анализа данных финансовых рынков методом временных рядов. С помощью пакета Neural Network можно применять нейронные сети к задачам нелинейного управления и финансового моделирования.

При решении задач оптимизации стоимости, надежности и качества в различных финансовых и экономических приложениях естественно использовать пакет Optimization, содержащий программы минимизации линейных и нелинейных функций.

Пакет GARCH включает современные методы экономических и финансовых исследований, в частности использует обобщенную составную модель ARMAX/GARCH для имитации, прогнозирования и оценки параметров временных рядов.

Использование MATLAB эффективно для предприятий, которые решают задачи анализа больших массивов данных. Поэтому первыми пользователями MATLAB для экономических целей стали Министерство финансов, Сбербанк, ряд крупных банков и компаний.

18 ГЛАВА Дополнения и примеры

В этой главе мы дадим несколько дополнений к материалу, изложенному в предыдущих главах, и рассмотрим ряд небольших иллюстративных задач. Это будут реализованные средствами MATLAB фрагменты задач, несколько превосходящие по объему кода те демонстрации команд, что были использованы для пояснения основ MATLAB в предыдущих главах:

- представлена технология создания Notebook-документов в редакторе MS Word, дающая возможность разработки «живых» электронных учебников. Для иллюстрации реализовано построение бифуркационной диаграммы знаменитого логистического отображения — одномерной динамической системы, обладающей хаотической динамикой;
- обсуждена проблема нахождения корней системы нелинейных уравнений, и, в частности, рассмотрено содержание широко известной библиотеки численных процедур NAG, дан пример реализации метода Ньютона для решения системы нелинейных уравнений. Для демонстрации метода нарисованы удивительные множества точек комплексной плоскости, из которых метод Ньютона сходится к заданным корням кубического полинома;
- дано применение графического интерфейса MATLAB, иллюстрирующее, что для интегрирования консервативных систем дифференциальных уравнений нужны специальные методы;
- представлены возможности MATLAB для трехмерной визуализации физических процессов: изображения полей скорости, линий тока, течения и т. д. При создании этих картин поясняются некоторые особенности управления характеристиками графики.

Детальное изложение этой информации невозможно из-за ограничений учебного курса, но знакомство с ней может быть полезным для читателя.

MATLAB в среде Word. Технология Notebook

В MATLAB имеется возможность проводить вычисления и визуализацию данных в среде редактора Microsoft Word. Данная технология называется Notebook и позволяет в едином документе (так называемой книге m-book) объединить текст, команды MATLAB и результаты их выполнения, включая графику. Это аналог Maple-документа (worksheet) и развитие команды-регистратора diary, записывающей в файл команды интерактивной сессии. Напомним, что в MATLAB 6 имеется специальное окно History Window, хранящее команды, запускавшиеся из командного окна.

Обзор команд Notebook

Дадим краткий обзор Notebook и рассмотрим характерные особенности этой технологии работы с MATLAB на примере решения простой задачи. Для подготовки файла (M-book) используется специальный шаблон, который поддерживает динамический обмен данными (интерфейс DDE) между MATLAB и редактором Word. Такой обмен обеспечивают специальные стили для текста и ячеек, макрокоманды для обработки данных из этих ячеек и дополнительное меню Notebook.

Предварительно следует запустить в командном окне MATLAB команду установки

```
» notebook -setup
```

Далее нужно выбрать версию редактора:

```
Choose your version of Microsoft Word:
[1] Microsoft Word for Windows 95 (Version 7.0)
[2] Microsoft Word 97
[3] Exit, making no changes

Microsoft Word Version:
```

Затем в диалоге указать соответствующие exe (winword.exe)- и dot (normal.dot)-файлы. При успешном выполнении этих простых операций в каталоге шаблонов MS Word появится новый шаблон m-book.

Для создания нового документа notebook нужно в меню **Файл** выбрать пункт **Создать...** и выбрать шаблон m-book. В меню **Файл** при этом появится команда **New m-book** и добавится меню **Notebook** (см. рис. 18.1) с пунктами, назначение которых поясняется далее.

Работа с MATLAB в книге m-book реализуется при помощи ячеек, в которые записываются команды системы. Чтобы завести ячейку, достаточно набрать текст команды и затем нажать комбинацию клавиш **Alt-D** или выбрать пункт **Define Input Cell** в меню **Notebook**. Ячейка ввода заключается в специальные скобки и выделяется специальным стилем. Команда в ячейке исполняется при нажатии комбинации клавиш **Ctrl-Enter** или выполнении пункта меню **Evaluate Cell**. Результаты вычислений, предупреждения и сообщения об ошибках помещаются в ячейки вывода, следующие за ячейками ввода. По умолчанию результат выполнения команды выводится синим цветом, а сообщения об ошибках — красным. Преобразовать ячейку в текст можно при помощи комбинации клавиш **Alt-U** или обратившись к пункту меню **Undefine Cells**.

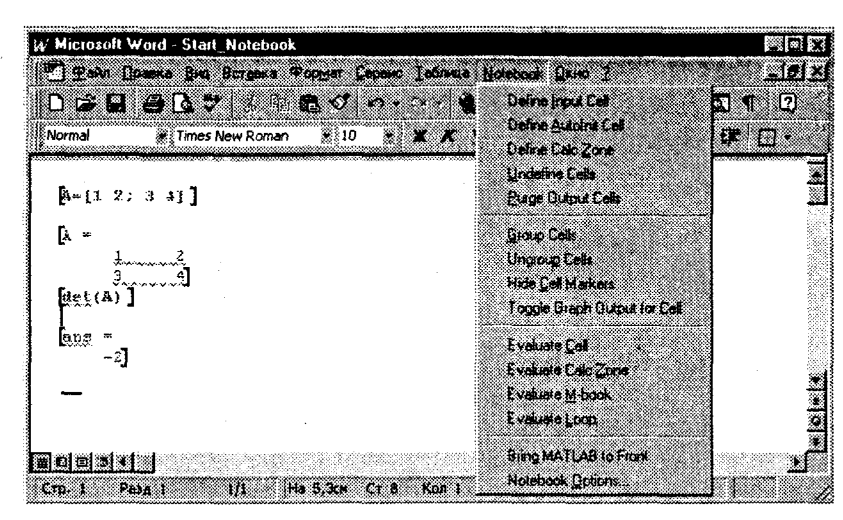


Рис. 18.1. Окно MS Word с меню Notebook

Для определения ячейки ввода нужно выделить все строки, которые занимает команда. Чтобы некоторые команды выполнялись при запуске книги m-book, можно определить эти ячейки как ячейки автостарта. Для этого имеется пункт меню Define AutoInit Cell.

Для одновременного выполнения нескольких команд можно определить группу ячеек. Для преобразования ряда ячеек в группу следует их выделить и использовать комбинацию клавиш Alt-G или пункт меню Group Cells. В группу ячеек не могут входить ячейки вывода и текст. Выделенные ячейки преобразуются в одну группу с общими для всех команд скобками. Для вычисления группы также используется комбинация клавиш Ctrl-Enter или пункт Evaluate Cell из меню Notebook. Результат выполнения размещается в одной ячейке вывода, располагающейся за группой. Так, в виде группы удобно организовывать последовательности команд построения и оформления графика.

Содержимое книги можно разбивать на секции (зоны), включающие текст, ячейки ввода и вывода. Для этого нужно выделить нужную область и выполнить пункт меню Define Calc Zone. Вычисление секции производится при помощи комбинации клавиш Alt-Enter или пункта Evaluate Calc Zone.

Если одновременно работать с несколькими книгами, то надо иметь в виду, что при этом используется одна рабочая область и запущен один процесс MATLAB.

Вычисление всей книги производится по команде меню Evaluate M-book или комбинации клавиш Alt-R. Имеется возможность организовать вычисление в цикле. Для этого нужно выделить и вызвать диалоговую панель (комбинация клавиш Alt-L или пункт меню Evaluate Loop). При помощи этой панели можно определить число повторов, запускать вычисления (Start) и приостанавливать их (Pause/Continue), регулировать скорость выполнения операций (кнопки Faster, Slower) или остановить расчет (Stop).

При помощи пункта Notebook Options можно изменить формат вывода данных (Numeric Format) и размеры графических окон (Figure Options). По умолчанию ри-

сунки помещаются в документе, для отмены надо снять флажок для параметра Embed Figures in M-book. Размеры рисунка можно зафиксировать при помощи полей Width и Height меню Notebook Options. Изменения вступают в силу после выполнения соответствующей ячейки. Кроме того, размер рисунка можно определить интерактивно. Для этого следует выделить рисунок, нажав левую клавишу мыши в любом месте рисунка, и затем установить нужный размер, перемещая угловую или срединную отметки. Пустое пространство вокруг рисунка можно удалить, если в процессе изменения размеров рисунка удерживать клавишу Shift. Флажок Use 16-Color Figures позволяет перейти от действующей по умолчанию палитры из 256 цветов к палитре из 16 цветов.

Для выделения команд ввода, результатов, сообщений об ошибках и текста используются различные стили. Назначения стилей по умолчанию приведены в табл. 18.1, размер шрифта везде 10 пунктов. Стили можно изменить так же, как это делается в редакторе Word. Надо помнить, что изменение стиля сказывается сразу на всем документе. При этом появится предложение изменить шаблон; и в результате сделанные коррекции отразятся во всех книгах.

Таблица 18.1. Стили по умолчанию

Стиль	Шрифт	Выделение	Цвет
Normal	Times New Roman		Черный
Input	Courier New	Bold	Зеленый
Output	Courier New		Синий
AutoInit	Courier New	Bold	Темно-синий
Error	Courier New	Bold	Красный

Когда расчеты проведены, то содержимое книги m-book может быть преобразовано в обычный документ Word. Чтобы превратить ячейку вывода в обычный текст, ее нужно выделить и выполнить пункт меню Undefine Cells или нажать Alt-U. При этом удаляются маркеры ячеек, а графика, числа и текст сохраняются. Чтобы удалить ячейку вывода, применяется пункт меню Purge Output Cells или комбинация клавиш Alt-P.

Бифуркационная диаграмма логистического отображения

В качестве примера документа Notebook рассмотрим построение бифуркационной диаграммы для классического примера динамической системы с хаотической динамикой — логистического отображения [42, 65]. Это отображение имеет единственный вещественный параметр b, и переменная берется из интервала [0,1]. Формула для расчета последовательных состояний из начальной точки имеет вид итерационного процесса (система с дискретным временем):

$$x_{n+1} = f(x_n), \quad n = 0,1,2,\ldots, \quad f(x) = bx(1-x) \cdot$$

При малых значениях параметра b<1 итерации из любых начальных точек стремятся к нулю — единственному стационарному решению. При переходе параметра b через 1 нуль теряет устойчивость и появляется новая устойчивая неподвижная точка, притягивающая все итерации из отрезка [0,1]. Такое возникновение нового режима и изменения устойчивости у существовавшего режима называется бифуркацией. Когда параметр b превышает число 3, рождается периодический режим из двух точек (новая бифуркация), и далее с ростом b данная система демонстрирует каскад удвоений — последовательность бифуркаций удвоения, потому что каждый новый режим состоит из удвоенного числа точек. Данный каскад приводит к формированию устойчивого бесконечного множества точек — странному аттрактору. При расчете новых итераций точки будто блуждают по отрезку [0,1], такое поведение называют стохастической или хаотической динамикой.

Алгоритм расчета бифуркационной диаграммы заключается в следующем. Для каждого значения параметра b из некоторого набора проводится расчет на установление. Затем выполняется достаточно большое число итераций, чтобы представить полученный при данном значении режим. Конечно, для периодических режимов (циклов), состоящих из малого числа точек, не требуется столько точек, сколько необходимо для представления хаотического странного аттрактора.

На рис. 18.2 приведено окно редактора Word с программой вычисления бифуркационной диаграммы для логистического отображения.

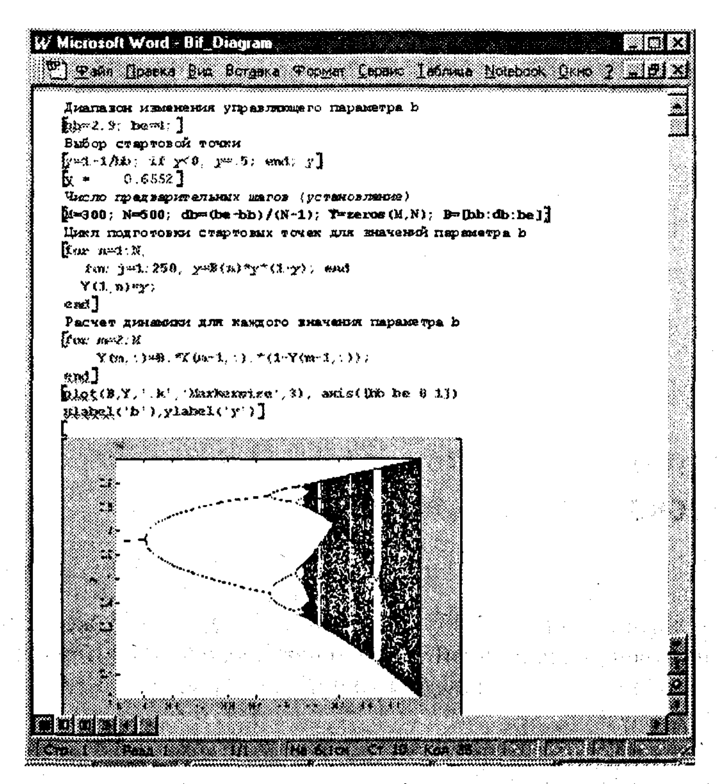


Рис. 18.2. Окно редактора Word с документом Notebook

Решение нелинейных уравнений

Решения системы нелинейных уравнений можно находить, используя стандартную функцию MATLAB для минимизации функции нескольких переменных fmins. Для этого можно написать функцию, вычисляющую, например, сумму квадратов левых (или правых) частей всех уравнений системы. Минимум такой неотрицательной функции будет достигаться на решениях исходной системы уравнений. Можно также воспользоваться для решения командой solve из пакета Symbolic Math Toolbox.

Наши учителя говорили, что трудно доверять численному решению, полученному одним методом. В практике каждого вычислителя найдутся подтверждающие это примеры. Поэтому вовсе не случайна избыточность пакета MATLAB, в котором имеется несколько методов решения одной задачи. В этом разделе опишем, как воспользоваться для решения алгебраической системы процедурами библиотеки NAG (Numerical Analysis Group) и как реализовать метод Ньютона в виде m-функции. Для иллюстрации разберем нахождение корней кубического полинома и построение карт начальных значений, из которых итерации сходятся к каждому корню.

Метод Ньютона описан во всех руководствах по численному анализу и численным методам, см., например, [Б75, К78, С84]. Пусть задано скалярное уравнение

$$f(x) = 0$$

В этом случае расчет приближений к корню ведется по формуле

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0,1,2\ldots$$

Для начала вычислений необходимо начальное приближение x_0, итерации продолжаются, пока разность $x_{n+1} - x_n$ не станет достаточно малой. Для системы нелинейных уравнений f по x есть векторы, а производная от f по x — матрица Якоби.

Библиотека NAG

Замечательная коллекция процедур численного анализа на Фортране поддерживается фирмой NAG Ltd. В MATLAB на основе этой библиотеки имеется пакет Toolbox NAG, куда входит более 250 процедур решения разнообразных задач. Для просмотра содержимого нужно обратиться к справке или выполнить команду

```
help NAG
```

Пакет состоит из следующих разделов (в скобках будем указывать префикс раздела):

- **вычисление нулей полиномов (c02);**
- **нахождение корней трансцендентных уравнений и систем (c05);**
- **вычисление рядов, преобразования Фурье и т. д. (c06);**
- **квадратуры для одномерных и многомерных интегралов (d01);**
- **методы решения обыкновенных дифференциальных уравнений (задачи с начальными данными и краевые задачи) (d02);**

- интегрирование уравнений в частных производных (d03);
- задачи интерполяции (e01);
- аппроксимация кривых и поверхностей (e02);
- задачи минимизации (e04);
- факторизация матриц (f01);
- вычисление собственных значений и векторов (f02);
- решение систем линейных алгебраических уравнений (f04);
- процедуры из пакета LAPACK (f07);
- обработка статистических данных (g01);
- вычисление корреляций и регрессионный анализ (g02) и (g03);
- генераторы случайных чисел (g05);
- методы непараметрической статистики (g08);
- анализ временных рядов (g13);
- методы сортировки (m01);
- аппроксимация специальных функций (s01), (s13)–(s15), (s17)–(s21).

Для справки по функциям библиотеки нужно знать имя интересующей нас функции, которое состоит из двух частей: префикса раздела (три буквы) и идентификатора (три буквы). Например, для получения справки по функции c05nbf нужно набрать

```
» help c05nbf
```

Решение системы нелинейных уравнений можно разыскивать при помощи функции c05nbf, которая требует только вычисления значений функций правой части. Кроме того, в пакете Toolbox NAG имеется функция c05pbf, для использования которой необходимо знать якобиан системы (матрицу первых производных от уравнений по переменным).

Обращение к функции c05nbf имеет вид:

```
[x,f,ifail] = c05nbf(fcn,x0,xtol,ifail)
```

Обязательными параметрами являются имя функции для вычисления правой части системы fcn и начальное приближение к решению x0. По умолчанию расчет ведется с точностью sqrt(eps), где eps есть константа MATLAB для используемого компьютера. Точность можно изменить, задав параметр xtol; целая переменная ifail по умолчанию равна -1. Для расчета итераций применяется вариант метода Ньютона с численным определением матрицы Якоби направленными разностями. Выходными параметрами являются найденное решение x, значение функции на решении f и переменная ошибки ifail.

Для определения корней подготовим функцию вычисления правой части в файле c05_f.m. Обратим внимание, что имя функции в заголовке может быть произвольным (например, fcn), а функция опознается по имени файла. Кроме того, в списке параметров должна присутствовать размерность вектора неизвестных x, даже если она не используется при вычислении значений функций:

```
» function [f,iflag]=fcn(n,x,iflag)
f=[x(1)*(x(1)^2-3*x(2)^2)+1, x(2)*(3*x(1)^2-x(2)^2)];
```

Затем обратимся к функции c05nbf с полным набором выходных параметров:

```
» [x,f,ifail]=c05nbf("c05_f",[1 1])
x =
  5.0000e-001
  8.6603e-001
f =
 -1.2992e-012
  3.5959e-014
ifail =
     0
```

В результате выполнения команды найдено одно из решений, нулевое значение переменной ifail подтверждает, что вычисления завершились нормально.

Обращение к функции c05pbf выглядит следующим образом:

```
[x,f,jac,ifail] = c05pbf(fcn,x0,xtol,ifail)
```

Здесь входные параметры те же, что и для функции c05nbf. К выходным параметрам добавилась матрица Якоби jac. Для итераций используется стратегия, сочетающая метод Ньютона с методом градиентного спуска. В функции fcn вычисляется правая часть системы и матрица Якоби. Для этого предусмотрена переменная flag, которая при значении 2 вычисляет значения матрицы Якоби, а при прочих значениях — правую часть системы:

```
function [f,jac,flag]=fcn(n,x,f,jac,ldfjac,flag)
%
if flag~=2
   f = [x(1)*(x(1)^2-3*x(2)^2)+1, x(2)*(3*x(1)^2-x(2)^2)];
else
   jac =[3*x(1)^2-3*x(2)^2  -6*x(1)*x(2); ...
         6*x(1)*x(2)         3*x(1)^2-3*x(2)^2];
end
```

Обращение к функции c05pbf с начальным значением x0=[-1 1] приводит к тому же результату, что и для функции c05nbf. Для вычисления другого решения изменим начальное приближение:

```
» x=c05pbf("c05_f_j",[-1 1]); x'
ans =
 -1.0000e+000  1.5681e-013
```

Пример функции для решения системы нелинейных уравнений методом Ньютона

Приведем пример программирования функции для нахождения корней системы нелинейных уравнений. Реализуем прямой метод Ньютона, который будет работать как для задачи с введенной матрицей Якоби, так и для случая численного определения матрицы Якоби. Далее представлен текст функции newtons вместе с необходимыми пояснениями.

В заголовке функции указан порядок следования входных и выходных параметров, комментарии объясняют их назначение:

```
function  [z0,n] = newtons(fun, z0, nmax, del, h, result)
% Newtons - решение системы нелинейных уравнений f(x)=0

% fun  - имя функции правой части
% z0   - начальное приближение
% nmax - предельное число итераций
% del  - точность метода Ньютона
% h    - шаг для вычисления якобиана численно
%  (отсутствие, 0 или [] - м. Якоби дана в функции fun)
% result - имя файла для записи итераций

% Пример вызова:
% [x,n]=newtons("c05_fj3",[1 i],10,1e-6,1e-3,'result.txt')
```

Далее идет анализ входных параметров и задание их значений в том случае, если при вводе они были опущены. Важным является пятый параметр — величина шага h. Если этот параметр отсутствует, на его месте находится пара квадратных скобок или он равен нулю, то этим указывается функции newtons, что вычисление матрицы Якоби содержится в функции вычисления уравнений fun. Если необходима запись результатов решения системы в файл, то в качестве третьего и четвертого параметров могут фигурировать пустые квадратные скобки:

```
if nargin<2, disp("Нет начального приближения"),return
end
if nargin<3 | isempty(nmax), nmax=9; end       % нет nmax
if nargin<4 | isempty(del), del=sqrt(eps);end  % нет del
if nargin<5,  h=sqrt(eps); end                 % нет h

if nargin<5 | isempty(h) | h==0,
     jacob=1; h=0;              % Матрица Якоби дана формулой
else jacob=0;                   % Матрица Якоби рассчитывается
end
```

Начальное приближение может быть задано вектором или матрицей. В расчетах будем использовать вектор-столбец, так что вначале производится соответствующее преобразование. После этого следует обращение к функции jaco, которая помещается в том же файле и предназначена для вычисления правой части и матрицы Якоби:

```
nz=length(z0(:)); z0=z0(:);        % Столбец
[fz0,jac]=jaco(fun,z0,nz,jacob,h);
```

Если указано имя файла (шестой параметр), то нужно приготовить переменные, в которые будет заноситься информация об итерациях и достигнутой точности (отклонении от нуля):

```
if nargin==6, z= z0'; R=norm(-fz0); end
```

Теперь запускается цикл для расчета итераций, где на каждом шаге метода Ньютона анализируется матрица Якоби. Вычисления прекращаются, если матрица Якоби содержит нечисловой параметр, бесконечность или если она вырождена:

```
for n=1:nmax
   detjac = det(jac);
   if ~any(isfinite(jac(:))) | ~isfinite(detjac),
       disp("Матрица Якоби плоха");     return
   elseif (detjac ~= 0),
        delta=-jac\fz0;   z0=z0+delta;
```

```
   else disp("Матрица Якоби необратима");  return
   end
   [fz0,jac]=jaco(fun,z0,nz,jacob,h);
   test=norm(fz0); tolrel=norm(delta)/(norm(z0)+eps);
   if nargin==6, z(n+1,:)=z0'; R=[R; test]; end
   if tolrel < del & test <= del,  converg = 1; break;
   elseif n==nmax,
    converg = 0;
    fprintf("\rНет сходимости%2.0f итераций\r\n",nmax);
   end
end
```

Выход из цикла происходит по достижении заданной точности или если будет достигнуто максимальное число итераций nmax. Результат последней итерации превращается в вектор-строку для использования в выходных параметрах:

```
z0=z0';
```

Если шестым параметром было указано имя файла для записи результатов, то происходит сохранение подготовленной информации о расчете:

```
if nargin > 5,
  [n,nz] = size(z); fid=fopen(result, "w");
  fprintf(fid,'Итерация  Норма  |  Решение \r\n');
  for k=1:n
     fprintf(fid, "%3.0f %12.2E", k, R(k));
     for m=1:nz,
       fprintf(fid,'%15.3E+i(%11.3E)',...
               real(z(k,m)),imag(z(k,m)));
     end
     fprintf(fid,'\r\n');
  end
  if ~converg
  fprintf(fid,'\r\nНет сходимости%2.0f итераций\r\n',nmax);
  end
  fclose(fid);
end
```

Функция jaco предназначена для вычисления значений функции fun и матрицы Якоби. Если в функции fun не содержится явных формул для матрицы Якоби, то производится расчет с использованием односторонних разностей:

```
function [fz,jac]=jaco(fun,z,nz,jacob,h)
if jacob,
   [fz,jac]=feval(fun,z,1);
else
   fz=feval(fun,z);
   for k = 1:nz
       de=zeros(nz,1); de(k)=h;
       jac(:,k)=(feval(fun,z+de)-fz)./h;
   end
end
```

Функцией jaco завершается файл newtons.m. Для решения конкретной системы нужно подготовить файл, в котором будет вычисляться вектор-функция и, если надо, матрица Якоби.

В качестве примера m-файла, задающего уравнения системы и матрицу Якоби, приведем функцию, позволяющую вычислять одновременно несколько корней

кубического полинома. В этом случае каждая компонента вектор-функции зависит только от соответствующей компоненты вектора неизвестных и матрица Якоби диагональная:

```
function [f,jac,flag]=fcn(z,flag)
if nargin<2 | isempty(flag) | ~flag,
   flag=0; jac=[];
end
f=z.^3+1;
if flag, jac=diag(3*z.^2); end
```

Представим результат вычисления сразу трех корней. Отметим, что такой способ возможен только для нахождения комплексных корней скалярного уравнения. Итерации производятся до тех пор, пока условие сходимости не выполнится для всего вектора решения:

```
» [x,n]=newtons("fcn",[1 i -i],20,[],[],'result.txt')
x =
  -1.0000             0.5000 - 0.8660i   0.5000 + 0.8660i
n =
    10
```

Этой функцией можно пользоваться и при вычислении одного корня. Проверим заодно, что получается, если число итераций невелико:

```
» [x,n]=newtons("c05_fj3",i,5,[],[],'result.txt');
Нет сходимости 5 итераций
```

Теперь посмотрим результаты, помещенные в файл result.txt:

```
Итерация  Норма  |  Решение
  1     1.41E+000     0.000E+000+i(-1.000E+000)
  2     5.97E-001     3.333E-001+i(-6.667E-001)
  3     3.31E-001     5.822E-001+i(-9.244E-001)
  4     2.73E-002     5.088E-001+i(-8.682E-001)
  5     2.44E-004     5.001E-001+i(-8.660E-001)
  6     1.98E-008     5.000E-001+i(-8.660E-001)

Нет сходимости 5 итераций
```

Видно, что для достижения точности и определения корня не хватило всего одной итерации.

В общем случае матрица Якоби не обязательно диагональная. Например, если записать кубический полином в виде двух уравнений относительно вещественной и мнимой частей неизвестной, то m-файл будет выглядеть следующим образом:

```
function [f,jac,flag]=fcn(x,flag)
   f = [x(1)*(x(1)^2-3*x(2)^2)+1, x(2)*(3*x(1)^2-x(2)^2)];
if flag
   jac = [3*x(1)^2-3*x(2)^2  -6*x(1)*x(2); ...
          6*x(1)*x(2)         3*x(1)^2-3*x(2)^2];
end
```

Если матрицу Якоби находить численно, то функция для определения корней кубического полинома будет иметь простейший вид:

```
function f=fcn(z)
f=z.^3+1;
```

Бассейны для корней кубического полинома

Рассмотрим кубическое уравнение $z^3+1=0$, имеющее один действительный и пару комплексно-сопряженных корней. В зависимости от начальной точки метода Ньютона будут сходиться к разным корням. Область начальных значений, из которых итерации метода Ньютона приводят к одному и тому же корню уравнения $f(x) = 0$, назовем бассейном корня. Для рассматриваемой задачи комплексная плоскость делится на три части по числу корней, и при этом границы бассейнов имеют фрактальную структуру.

Подготовим программу рисования бассейнов для корней кубического уравнения. Будем окрашивать бассейн каждого корня своим цветом, а для большей изобразительности используем изменение интенсивности цвета в зависимости от числа итераций, которые пришлось выполнить для нахождения корня с заданной точностью. Быстрой сходимости к решению будет отвечать темный тон, и интенсивность будет убывать по мере увеличения числа итераций.

Подготовим данные для расчета бассейнов в области [-3,1]×[-1.5,1.5] (параметры a1,a2,b1,b2) с числом точек 400×300. Введем массив начальных точек Z0, массив результатов ZZ и массив для хранения числа потребовавшихся итераций LL:

```
» a1=-3; a2=1; b1=-1.5; b2=-b1; n=200; m=150;
hx=(a2-a1)/(2*n-1); hy=(b2-b1)/(2*m-1);
[X,Y]=meshgrid(a1:hx:a2,b1:hy:b2);
Z0=X+i*Y; e1=.001; LL=zeros(2*m,2*n); ZZ=LL;
```

Запустим вычисления методом Ньютона для всех точек из массива Z0. Для данного примера не будем пользоваться обращением к приведенной выше функции newtons, а запишем явную формулу для вычисления итераций:

```
» for k=1:length(Z0(:)),
   z0=Z0(k); z=z0+1;
   while abs(z-z0)>e1,
     LL(k)=LL(k)+1; z0=z; z=(2*z^3-1)/(3*z^2);
   end, ZZ(k)=z;
  end
```

Для подготовки рисунка преобразуем данные, смягчив логарифмированием отличия по числу потребовавшихся итераций и масштабировав элементы массива LL от нуля до единицы:

```
» LL=log(log(LL)+1); Lmin=min(min(LL)); Lmax=max(max(LL));
LL=(LL-Lmin)/(Lmax-Lmin);
```

Подготовим палитру из 66 интенсивностей серого, удалив темные и близкие к белому тона:

```
» nc=66; cc=ones(nc,3);
for k=1:nc; cc(k,:)=.45+k/nc*.45; end;
colormap(cc);
```

Вычислим массив С, используя для каждого корня свой диапазон интенсивности серого тона. Обратим внимание, насколько просто в MATLAB произвести разделение начальных точек по корням, к которым привели итерации метода Ньютона.

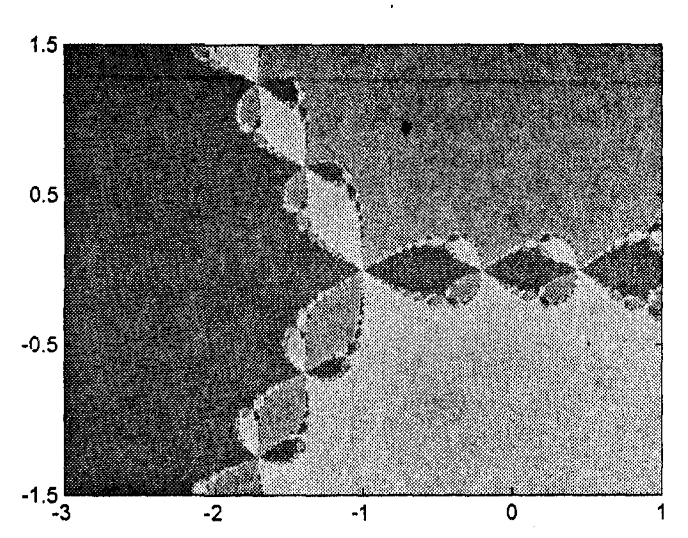


Рис. 18.3. Бассейны корней. Серая палитра с монотонным убыванием интенсивности

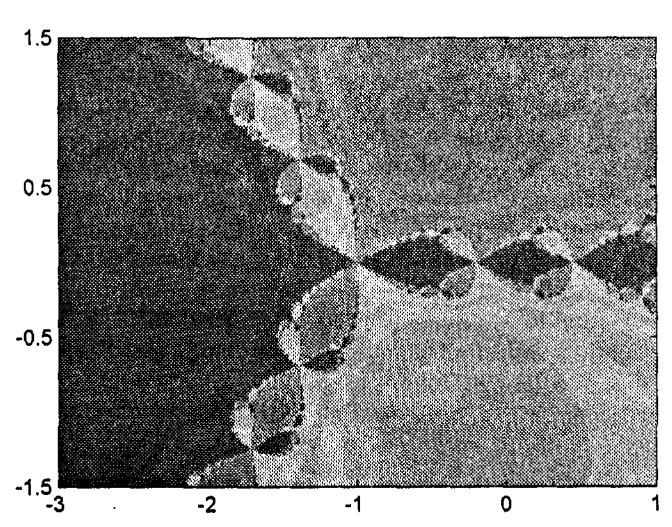


Рис. 18.4. Серая палитра с вариациями интенсивности

Затем обратимся к команде image для вывода изображения, подготовленного массивом C. На рис. 18.3 приведен результат.

```
» C=fix(1+(LL+(imag(ZZ)>e1)+2*(imag(ZZ)<-e1))*nc/3);
image(X(1,:),Y(:,1),C), axis image, axis([a1 a2 b1 b2])
```

Усовершенствуем рисунок, изменив палитру так, чтобы интенсивность каждого второго тона чуть возросла (напомним, что нуль отвечает черному цвету, а единица — белому). Для этого достаточно выполнить следующую команду:

```
» for k=2:2:nc, cc(k,:)=cc(k,:)*.95;end, colormap(cc);
```

Полученное в результате изображение приведено на рис. 18.4.

Изменим параметры области так, чтобы рассмотреть распределение точек в окрестности нуля:

```
» a1=-1; a2=.5; b1=-.25; b2=-b1; n=240; m=80;
```

Полученное изображение приведено на рис. 18.5. Хорошо видно, что множество начальных точек, из которых метод Ньютона сходится к данному корню, имеет фрактальный характер.

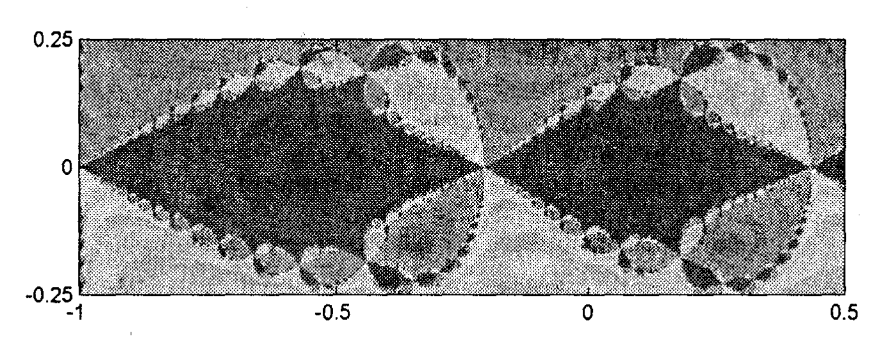


Рис. 18.5. Фрагмент картины бассейнов корней

Для сравнения приведем результаты вычисления бассейнов при помощи процедур пакета NAG. В левой части рис. 18.6 дана карта бассейна для корней кубического полинома, рассчитанных при помощи функции c05nbf, а в правой части — соответственно для c05pbf. Каждому корню отвечает своя интенсивность серого, а белым цветом даны точки, итерации из которых закончились сообщением об аварийном завершении работы процедур. Видно, что алгоритмы из библиотеки NAG дают картины, отличные от полученных прямым методом Ньютона, см. рис. 18.3. В отличиие от рис. 18.3 центром сгущения областей является начало координат. Таким образом, результаты численного исследования бассейнов кубического полинома существенно зависят от используемых методов.

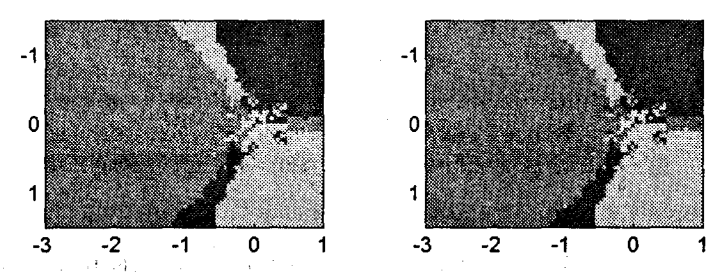


Рис. 18.6. Бассейны корней для процедур из библиотеки NAG

Для получения цветного рисунка можно воспользоваться одной из многих палитр MATLAB, см. главу 14 «Графика MATLAB», или подготовить собственную. Например, палитру из трех цветов (синий, зеленый и красный) с оттенками можно создать при помощи следующих команд:

```
» nc=66; nc1=11; cc=zeros(nc,3);
for k=1:nc1,
```

```
    cc(k,3)=.5+k/ncl*.5;
    cc(k+ncl,3)=1; cc(k+ncl,2)=.4+k/ncl*.6;
    cc(k+2*ncl,2)=.4+k/ncl*.3;
    cc(k+3*ncl,2)=0.7+k/ncl*.3;
    cc(k+4*ncl,1)=.6+k/ncl*.4;
    cc(k+5*ncl,2)=.5+k/ncl*.4; cc(k+5*ncl,1)=1;
end
```

Разработка приложения с GUI

Опишем последовательность действий, необходимую для создания интерактивного графического приложения. Представим пример «ручного» программирования графического интерфейса и приведем тексты на языке MATLAB, реализующие соответствующие конструкции. Конечно, используя визуальное проектирование при помощи Guide, решать подобные задачи проще, но вместе с тем полезно знать, как именно устроены команды, реализующие поддержку графики, и опыт такого рода может помочь при отладке графических приложений и квалифицированной доводке сложных программ.

Материалом для данного примера послужит тема сохранения интеграла энергии при численном интегрировании системы дифференциальных уравнений для консервативной механической задачи. Для простоты ограничимся системой второго порядка, p означает импульс, а q — координату. Гамильтониан системы имеет вид

$$H=\frac{p^2}{2}+\frac{k^2 q^2}{2}$$

Здесь k — коэффициент (безразмерная частота). Отвечающая гамильтониану система дифференциальных уравнений записывается следующим образом:

$$q_t=p$$

$$p_t=-k^2 q$$

Данная система автономных уравнений (время t не входит явно) может быть записана в векторном виде. Введем вектор неизвестных $Y=(q,p)$, тогда вектор правой части имеет вид $F(Y)=(p,-k^2q)$, а общий вид системы дается формулой

$$Y_t=\mathrm{F}(Y)$$

Для интегрирования этой задачи воспользуемся тремя простыми методами [73] и выпишем соответствующие формулы:

- Явный метод Эйлера

$$Y_{n+1}=Y_n+h\,\mathrm{F}(Y_n)$$

- Неявный метод Эйлера

$$Y_{n+1}=Y_n+h\,\mathrm{F}(Y_{n+1})$$

- Метод средней точки

$$Y_{n+1} = Y_n + h\,F\left(\frac{Y_n + Y_{n+1}}{2}\right)$$

Здесь индекс n используется для обозначения решения в t_n-й момент времени. Явный метод Эйлера дает расчетную формулу для вычисления решения на каждом шаге, а два других метода являются неявными, и для их реализации в общем случае необходимо решать нелинейные системы уравнений. Для рассматриваемого квадратичного гамильтониана эти методы можно записать в виде явных расчетных формул.

Наша цель — реализовать указанные методы для гамильтоновой системы и изучить преобразования фазовых объемов, получаемые этими методами. Результаты иллюстрируют следующий эффект: при использовании методов, не являющихся симплектическими, фазовый объем гамильтоновой системы не сохраняется [73].

Подготовим ряд функций для графического оформления окна, расчета по формулам и вывода информации. В файле Symp_menu.m находится основная функция, содержащая описание глобальных переменных и элементов оформления. При запуске функции Symp_menu вызывается окно, в котором создается поле для фазового портрета, кнопки Go и Clear, список с названиями методов и текстовая надпись «Гармонический осциллятор».

Для работы с графическими объектами необходимы дескрипторы, получающиеся в результате запуска графических команд, см. главу 14 «Графика MATLAB». Все дескрипторы объявлены глобальными переменными, а их имена начинаются с буквы h. Дескрипторы hBut1 и hBut2 связаны с кнопками, дескрипторы hCheck, hLis и hText используются соответственно для нанесения (удаления) сетки на графике, реализации выбора элемента из списка и вывода надписи. Дескриптор hEdit используется для ввода числа в редактируемое текстовое поле. Переменная NStep (число шагов) также является глобальной, и ее начальное значение задается перед вызовом функции Symp_menu. Для размещения графических элементов в окне используем переменные xx и yy, задающие размер поля вывода, dx и dy, определяющие величину смещения по горизонтали и вертикали соответственно:

```
function Symp_menu
% menu
global NStep hFig hAxes hEdit hLis hBut1 hBut2 hText hCheck
xx=300; yy=150; dx=20; dy=20;

hFig=figure;
set(hFig,'Position',[yy yy xx+12*dx yy+5*dy])
hAxes=("Parent",hFig,'Color',[1 1 1], ...
   "Units",'points','Position',[dx dy xx yy])

hBut1=uicontrol(hFig,'Style','pushbutton', ...
   "String",'Go', "Position",[xx+7*dx 2*dy dx dy],...
   "Callback",'Symp_Go');
hBut2=uicontrol(hFig,'Style','pushbutton', ...
   "String",'Clear', "Position",[xx+9*dx 2*dy 2*dx dy],...
   "Callback",'Symp_Clear');
```

```
hCheck=uicontrol(hFig,'Style','checkbox', ...
   "String",'Grid','Position',[xx+8*dx 4*dy 2*dx dy],...
   "HorizontalAlignment",'left');
hList=uicontrol(hFig,'Style','listbox', ...
   "String",{"Explicit Euler",'Midpoint rule',...
   "Implicit Euler"},'TooltipString','Method', ...
   "Background",'white', "HorizontalAlignment",'left', ...
   "Position",[xx+7*dx 8*dy 5*dx 7/2*dy]);
hText=uicontrol(hFig,'Style','text', "String",'Steps:',...
   "Backgroundcolor",[.75 .75 .75],
   "HorizontalAlignment",'left', ...
   "Position",[xx+7*dx 5.8*dy 4.4*dx 1.2*dy]);...
hEdit=uicontrol(hFig,'Style','edit','String',StrEdit, ...
   "Background",'white', "HorizontalAlignment",'left', ...
   "TooltipString",'Positive Integer', ...
   "Position",[xx+9*dx 6*dy 2*dx dy]);
```

Напомним назначение использованных в графических командах параметров: Position определяет местоположение и размер графического объекта, Style — тип управляющего элемента, String — текстовую надпись, Callback — имя вызываемой функции, Backgroundcolor — цвет фона, HorizontalAlignment — стиль выравнивания по горизонтали, TooltipString — текст всплывающей подсказки. Имена параметров можно сокращать до тех пор, пока это позволяет однозначно их идентифицировать, для получения списка параметров достаточно выполнить команду get(H), где H — дескриптор нужной команды.

При каждом нажатии кнопки **Clear** вызывается функция очистки рисунка. Соответствующий файл **Symp_Clear.m** содержит строку описания глобальных переменных и обращение к команде cla:

```
function Symp_Clear
% Clear - очистим рисунок
global NStep hFig hAxes hEdit hLis hBut1 hBut2 hText hCheck
axes(hAxes)
cla
```

При нажатии кнопки **Go** вызывается функция Symp_Go, проводящая вычисления и рисование динамики изменения площади начальных данных на фазовой плоскости. Файл **Symp_Go.m** содержит описание глобальных переменных, команды подготовки данных и комментарии, которые помогают понять назначение выполняемых операций. Вычисления по формулам Эйлера с постоянным шагом реализованы без вызова функций и обращения к функции вычисления правой части системы дифференциальных уравнений:

```
function Symp_Go
% Go - Гармонический осциллятор и методы Эйлера.
% Демонстрация изменения фазового объема.

global NStep hFig hAxes hLis hBut1 hBut2 hText hCheck

% Импульсы и координаты
p=-2:.2:2; q=-1:.1:1; k=(sqrt(5)+1)/2;
% Массив значений гамильтониана
[pp,qq]=meshgrid(p,q); H=(pp.^2/2+k^2*qq.^2)/2;
% Оси
```

```
hold on, axis([p(1) p(end) q(1) q(end)])
% Начальная окружность
te=-pi:pi/12:pi; x0=1+.12*cos(te); y0=.12*sin(te);
h=pi/8/k;
% Число шагов
NStep=str2num(get(hEdit,'String'));
% Линии уровня
contour(p,q,H,9,'c');

% Сетка
if get(hCheck,'Value'), grid on, else grid off, end

% Выбор метода
index=get(hLis,'Value'); cellAr=get(hLis,'String');
meth=cellAr(index);

switch char(meth)
   case "Explicit Euler"  % Явный метод Эйлера
   x=x0; y=y0;
   for j=1:NStep
      fill(x,y,'y')
      x1=x; y1=y; x=x1-k^2*h*y1; y=y1+h*x1;
   end
   case "Midpoint rule" % Метод средней точки
   h=pi/8/k;
   x=x0; y=y0; d=1/(1+k^2*h^2/4);
   for j=1:NStep
      fill(x,y,'c')
      x1=x; y1=y; x=((1-h^2*k^2/4)*x1-k^2*h*y1)*d;
                  y=((1-h^2*k^2/4)*y1+h*x1)*d;
   end
   case "Implicit Euler"  % Неявный метод Эйлера
   x=x0; y=y0; d=1/(1+k^2*h^2);
   for j=1:NStep
      fill(x,y,'r');
      x1=x; y1=y; x=(x1-k^2*h*y1)*d; y=(y1+h*x1)*d;
   end
end   % end switch

% Начальная окружность
fill(x0,y0,'b')
```

Для запуска подготовленного примера подготовим файл Symp_demo с описанием глобальных переменных, заданием числа шагов NStep и обращением к функции Symp_menu:

```
global NStep hFig hAxes hEdit hLis hBut1 hBut2 hText hCheck
NStep=16;
Symp_menu
```

Теперь наберем в командной строке

```
» Symp_demo
```

Результат запуска представлен на рис. 18.7. Начальные данные для каждого расчета размещаются на окружности (темный круг на рис. 18.7). Явному методу Эйлера отвечает увеличение фазового объема, неявному — уменьшение, и только сим-

плектический метод средней точки сохраняет начальную площадь. Для явного метода Эйлера интеграл энергии возрастает, для неявного — убывает, а для метода «средней точки» — имеют место осцилляции интеграла.

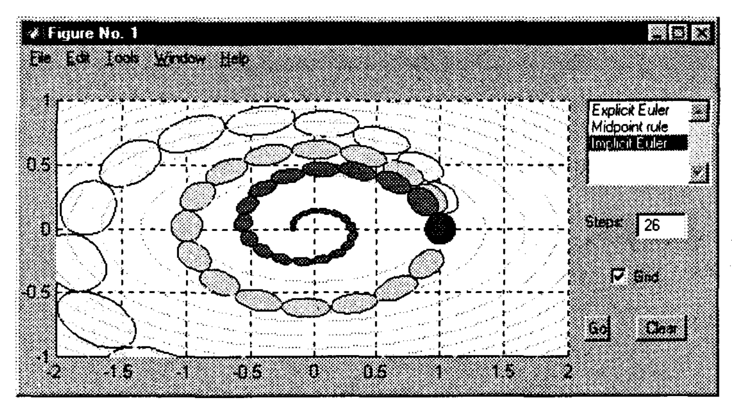


Рис. 18.7. Демонстрации изменения фазовых объемов в зависимости от метода интегрирования

Трехмерная визуализация функций и векторных полей

Графические команды MATLAB являются не только мощным инструментом визуализации математических результатов, но и эффективным аппаратом исследования. Здесь мы проиллюстрируем работу команд, предназначенных для визуализации функций и трехмерных векторных полей в трехмерном пространстве, в частности течений жидкости. Явное представление на экране структуры течений и их характеристик часто позволяет понять свойства течений и помочь при решении многих задач. Кроме того, этот пример демонстрирует, как можно совмещать на одном рисунке различные графические объекты.

В этом разделе использованы некоторые графические команды, которые не были описаны ранее в книге. Перечислим их. Поверхность постоянного значения `val` функции трех переменных, заданную значениями `V` в узлах с координатами `X,Y,Z`, можно построить командой `isosurface(X,Y,Z,V,val)`, а для вывода линий уровня этой функции на поверхности, заданной узлами `sx`, `sy`, `sz`, используется команда `contourslice(X,Y,Z,V,sx,sy,sz)`.

Для визуализации трехмерных векторных полей и течений в MATLAB имеются следующие команды:

- `coneplot` — предназначена для изображения векторного поля при помощи конусов или стрелок;
- `streamline` — выводит линии тока;
- `streamribbon` — иллюстрирует направление потока вместе с его вращением;
- `streamtube` — изображает поток в виде трубок, ширина которых зависит от значения дивергенции векторного поля.

Обязательными параметрами всех этих команд являются: X, Y, Z — массивы координат узлов, в которых задано векторное поле, U, V, W — значения компонент векторного поля в этих узлах и sx, sy, sz — координаты узлов для вывода конусов (стрелок) или начальных точек для построения линий тока.

Действие перечисленных команд мы будем демонстрировать на примере векторного поля с компонентами (v_x, v_y, v_z), определяющего течение сжимаемой жидкости в трехмерном пространстве. Динамика жидкой частицы в этом поле исследовалась в работе [73], где было показано, что линии тока формируют очень сложную структуру течения. Один из частных случаев рассматриваемого поля имеет вид [73]:

$$v_x = \varepsilon \sin(z) + 4\frac{\sinh(z)\cdot\sin(x)}{\cosh^3(z)} - 2\frac{\sin(y)}{\cosh^2(z)}$$

$$v_y = \varepsilon \cos(z) + 4\frac{\sinh(z)\cdot\sin(y)}{\cosh^3(z)} + 2\frac{\sin(x)}{\cosh^2(z)}$$

$$v_z = 2\frac{\cos(x)+\cos(y)}{\cosh^2(z)}$$

Здесь x, y, z — координаты в трехмерном пространстве, а ε — параметр. Отметим, что изучаемое векторное поле периодично по переменным x и y с периодом 2π, а потому при рассмотрении его структуры достаточно ограничиться интервалом $[-\pi, \pi]$ по этим переменным.

В первую очередь в виде m-файла опишем функцию vel, вычисляющую компоненты векторного поля в узлах прямоугольной сетки. Входными параметрами процедуры являются матрицы X, Y и Z, содержащие координаты узлов и значение параметра ε. Результатом выполнения процедуры будут три матрицы U, V и W, определяющие соответственно три компоненты векторного поля в узлах сетки:

```
function [U,V,W]=vel(X,Y,Z,epsilon)
U = epsilon*sin(Z)+4.*sinh(Z).*sin(X)./(cosh(Z).^3)-...
  2.*sin(Y)./(cosh(Z).^2);
V = epsilon*cos(Z)+4.*sinh(Z).*sin(Y)./(cosh(Z).^3)+...
  2.*sin(X)./(cosh(Z).^2);
W=(2.*cos(X)+2.*cos(Y))./(cosh(Z).^2);
```

Отметим, что при вычислении компонент вектора используются знаки поэлементных математических операций (то есть арифметические знаки с предшествующей точкой) для расчета вектора скорости для массива точек.

Чтобы создать рисунок, сгенерируем сетку для аппроксимации векторного поля командой meshgrid и вычислим значения компонент вектора в этих узлах при $\varepsilon=0$:

```
» [X Y Z]=meshgrid(-pi:pi/20:pi,-pi:pi/20:pi,-1.55:0.1:1.55);
[U V W]=vel(X,Y,Z,0);
```

Следующие команды находят минимумы и максимумы значений трех координат, вычисляют длину вектора в узлах сетки и присваивают их значение переменной velos, а две последние команды создают новый рисунок и включают режим сохранения объектов на рисунке:

```
» xmin=min(X(:)); xmax=max(X(:)); ymin=min(Y(:));
```

```
ymax=max(Y(:)); zmin=min(Z(:)); zmax=max(Z(:));
velos=sqrt(U.^2+V.^2+W.^2); figure; hold on
```

Теперь все готово для изображения структуры векторного поля. Построим при помощи команды slice на трех плоскостях *y=pi, z=-1.5* и *x=-pi* картины плотности функции velos и введем дескрипторы h1, h2 и h3 для этих изображений. Параметр cubic указывает на то, что для приближения функции используется кубическая интерполяция. Для каждой плоскости команда set задает режим интерполирования цвета по значениям функции:

```
» h1=slice(X,Y,Z,velos,[],pi,[],'cubic');
set(h1,'FaceColor','interp','EdgeColor','none');
h2=slice(X,Y,Z,velos,[],[],[-1.5],'cubic');
set(h2,'FaceColor','interp','EdgeColor','none');
h3=slice(X,Y,Z,velos,-pi,[],[],'cubic');
set(h3,'FaceColor','interp','EdgeColor','none');
```

Команда streamslice изображает линии тока векторной функции, определенной значениями компонент вектора [U,V,W] в узлах сетки на заданных плоскостях. В примере выбраны три плоскости: *z=-1, z=0* и *z=1.5*. После построения линий тока присвоим им черный цвет командой set:

```
» h2s=streamslice(X,Y,Z,U,V,W,[],[],[-1.0 0.0 1.5]);
set(h2s,'Color','k');
```

Для представления векторного поля можно использовать не только плоские сечения, но и сложные поверхности. Эти поверхности сначала нужно определить координатами узлов, а затем обратиться к нужной команде, указав в качестве параметров переменные, содержащие эти координаты. Например, зададим командой surf синусоидальную поверхность, а затем повернем ее командой rotate. После этого преобразованные координаты узлов присвоим переменным xd, yd и zd, а саму поверхность удалим командой delete. Наконец, введем на синусоидальной поверхности изображение плотности функции velos (дескриптор h4):

```
» hs=surf(linspace(xmin,xmax,101),linspace(ymin,ymax,101),...
   -0.5*sin(2*meshgrid(zmin:(zmax-zmin)/100:zmax)));
rotate(hs,[0,1,1],15);
xd=get(hs,'XData'); yd=get(hs,'YData'); zd=get(hs,'ZData');
delete(hs);
h4=slice(X,Y,Z,velos,xd,yd,zd,'cubic');
set(h4,'FaceColor','interp','EdgeColor','none');
```

Следующая группа команд завершает оформление рисунка. Команда daspect устанавливает пропорции по осям, в частности увеличенный в два раза размер по оси z. Команда view задает угол взгляда на рисунок. Затем определяется заголовок рисунка, надписи по осям координат и область вывода. Три последние команды соответственно задают палитру рисунка (оттенки серого), шкалу значений функции vel и осветление рисунка:

```
» daspect([1 1 0.5]); view(25,15); title("CABC-течение");
xlabel("X"); ylabel("Y"); zlabel("Z");
axis([-pi pi -pi pi -1.5 1.5]);
colormap gray; colorbar; brighten(0.5);
```

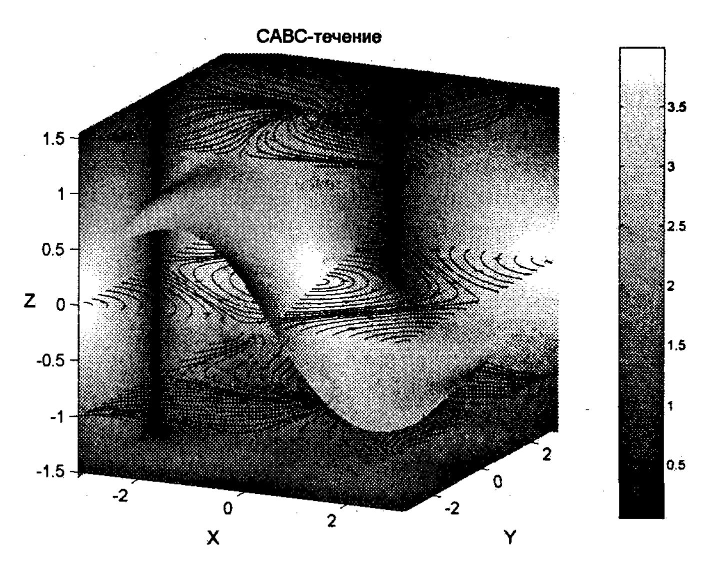


Рис. 18.8. Три плоскости с линиями тока вместе с тремя плоскими и одним синусоидальным сечением, раскрашенными в зависимости от значений функции

Результат работы описанных команд представлен на рис. 18.8.

Создадим новое окно и для большей наглядности свойств течения определим новую сетку, выбрав интервал величиной в два периода по осям x и y:

```
» figure
[X Y Z]=meshgrid(0:pi/10:4*pi,0:pi/10:4*pi,-pi:pi/10:pi);
```

Теперь вычислим компоненты векторного поля и значение модуля скорости в узлах сетки:

```
» [U V W]=vel(X,Y,Z,0);
velos=sqrt(U.^2+V.^2+W.^2);
```

На четырех граничных плоскостях с помощью команды contourslice изобразим линии уровня модуля скорости velos и зададим их толщину и цвет (черный):

```
» hcont=contourslice(X,Y,Z,velos,[0,4*pi,4*pi],4*pi,-pi,8);
set(hcont,'EdgeColor',[0 0 0],'LineWidth',0.5);
```

Для изображения линий тока течения подготовим набор начальных точек:

```
» [sx sy sz]=meshgrid(pi,0:pi/4:5*pi/4,-pi/2:pi/4:pi/2);
```

Теперь для вывода линий тока в виде трубок с толщиной, которая зависит от дивергенции векторного поля, обратимся к команде streamtube. Дескриптор этого объекта есть переменная htubes, командой set определены цвета поверхности трубки, сетки на ней и освещенность:

```
» htubes=streamtube(X,Y,Z,U,V,W,sx,sy,sz);
  set(htubes,'EdgeColor','none','FaceColor','r',...
   "AmbientStrength",0.5);
```

Для завершения рисунка определим ракурс, заголовок, маркировку осей координат и источник света.

```
» view(60.35); title("САВС-течение");
xlabel("X"); ylabel("Y"); zlabel("Z");
axis tight; axis equal; camlight left
```

Результат приведен на рис. 18.9.

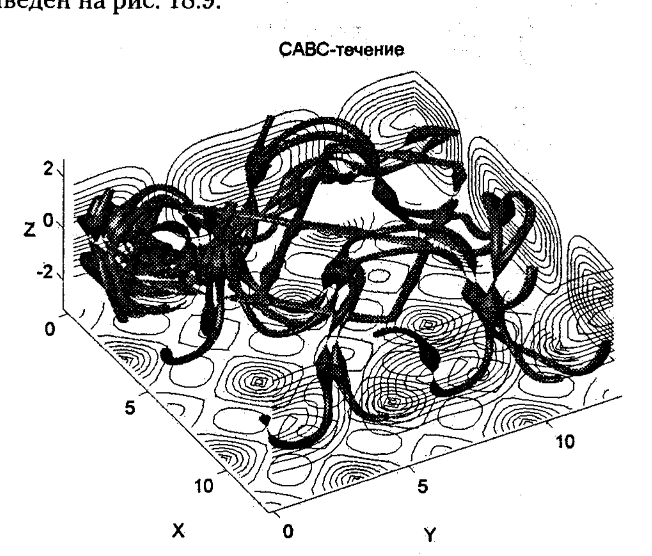


Рис. 18.9. Изображение линий тока в виде трубок с толщиной, зависящей от дивергенции векторного поля

На следующем рисунке для вывода линий тока в виде лент (команда streamribbon) воспользуемся уже вычисленными сеткой, векторным полем и начальными точками. Командой set определим темно-серый цвет лент, уровень освещенности и отменим штриховку на лентах:

```
» figure;
hribb=streamribbon(X.Y.Z.U.V.W.sx.sy.sz.0.3);
set(hribb.'EdgeColor'.'none'.'FaceColor'....
   [0.2 0.2 0.2].'AmbientStrength'.0.5);
```

Теперь включим режим сохранения объектов на рисунке и определим еще один набор точек. Командой plot3 выведем этот набор в виде звезд черного цвета:

```
» hold on
[sx1 sy1 sz1]=...
   meshgrid(3*pi:pi/6:23*pi/6.2*pi.-pi/2:pi/4:pi/2);
plot3(sx1(:).sy1(:).sz1(:).'*k');
```

Используем определенные массивами sx1, sy1 и sz1 точки в качестве начальных для построения обычных линий тока (команда streamline) и зададим для них черный цвет:

```
» h=streamline(X.Y.Z.U.V.W.sx1.sy1.sz1);
  set(h. "Color". "black");
```

Закончим оформление очередного рисунка заданием характеристик осей координат, определением угла взгляда, положения и типа источника подсветки:

```
» view(60.35); axis tight; axis equal; grid on;
camlight(45.45.'infinit'); lighting flat;
```

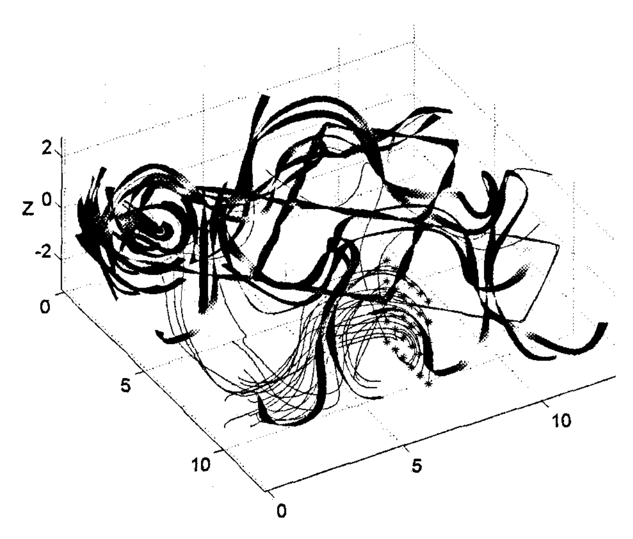


Рис. 18.10. Линии тока в виде лент и обычные линии тока

Полученное изображение линий тока представлено на рис. 18.10.

На заключительном рисунке данного раздела изобразим структуру и характеристики векторного поля в окрестности точки $x=\pi$, $y=0$, $z=0$ при $\varepsilon=0.1$. При $\varepsilon=0$ эта точка является особой точкой векторного поля, то есть в ней все компоненты вектора обращаются в ноль, а при изменении параметра в ее окрестности имеет место хаотическое движение жидкой частицы ([78]). Убедимся в том, что при $\varepsilon=0$ точка является особой:

```
» [v1,v2,v3]=vel(pi,0,0,0);
fprintf("%7.5f %7.5f %7.5f \n",v1,v2,v3)
0.00000 0.00000 0.00000
```

Теперь создадим рисунок, определим более подробную сетку и вычислим в ее узлах компоненты векторного поля при $\varepsilon=0.1$:

```
» figure;
  [X Y Z]=meshgrid(4*pi/5:pi/50:6*pi/5,-pi/5:pi/50:pi/5,...
              -pi/10:pi/100:pi/10);
  [U V W]=vel(X,Y,Z,0.1);
```

Переменной `vc` присвоим значение угловой скорости вихря. Эта величина вычисляется, если при обращении к этой команде указан один выходной параметр. При присвоении результата набору из трех переменных (например, `[vx, vy, vz] = =curl(x,y,z,u,v,w)`) в этих переменных будут находиться компоненты вектора вихря:

```
» vc=curl(X,Y,Z,U,V,W);
```

Далее с помощью команды `isosurface` выведем поверхность, на которой значение `vc` равно 1.5. Затем определим желтый цвет поверхности и отменим вывод сетки:

```
» h=patch(isosurface(X,Y,Z,vc,1.5));
set(h,'FaceColor','yellow','EdgeColor','none');
```

Теперь с помощью составной команды окрасим ограничивающие построенную поверхность плоскости цветом, зависящим от значения переменной vc. Отметим, что окрашиваются только те участки, где величина vc меньше указанного значения (в примере 1.5):

```
» hcap=patch(isocaps(X,Y,Z,vc,1.5),'FaceColor','interp'....
        "EdgeColor",'none');
```

Далее перейдем к изображению векторного поля конусами различной величины, которая зависит от длины вектора в точке. Сформируем сначала набор конусов на равномерной сетке, для чего определим эту сетку и обратимся к команде coneplot. Конусы этого набора будут окрашены в красный цвет:

```
» xr=linspace(min(X(:)),max(X(:)),6);
yr=linspace(min(Y(:)),max(Y(:)),6);
zr=linspace(min(Z(:)),max(Z(:)),4);
[cx cy cz]=meshgrid(xr,yr,zr);
h1=coneplot(X,Y,Z,U,V,W,cx,cy,cz,2);
set(h1,'FaceColor','red','EdgeColor','none');
```

Переменной velos присвоим сумму первых двух компонент векторного поля:

```
velos=U+V;
```

Второй набор конусов разместим на криволинейной поверхности, определяемой нулевым значением velos. Вычислим с помощью isosurface требуемую поверхность. Результатом этой команды будет некоторое количество вершин и граней, аппроксимирующих поверхность, а командой reducepatch оставим только десять процентов этих объектов. Второй набор конусов синего цвета разместим в полученных вершинах:

```
[f v]=reducepatch(isosurface(X,Y,Z,velos,0),0.1);
h2=coneplot(X,Y,Z,U,V,W,v(:,1),v(:,2),v(:,3),2,'cubic');
set(h2,'FaceColor','blue','EdgeColor','none');
```

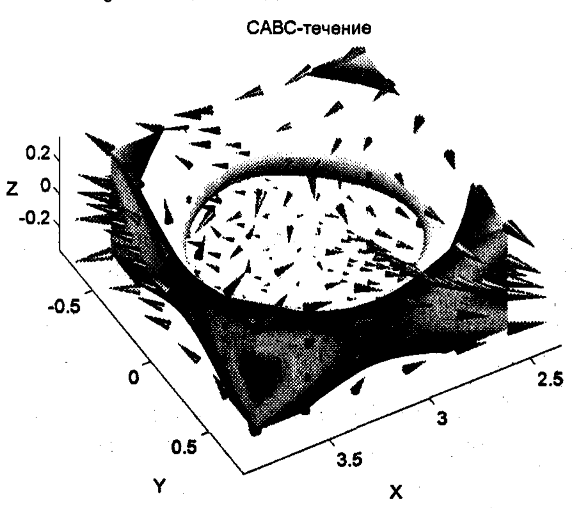


Рис. 18.11. Структура векторного поля в окрестности точки x=π, y=0, z=0 и поверхность постоянного значения угловой скорости изменения вектора вихря

Наконец, аналогично предыдущим, закончим оформление последнего рисунка этого раздела:

```
title("CABC-течение"); daspect([1 1 1]);
xlabel("X"); ylabel("Y"); zlabel("Z"); view(150,45)
axis tight; axis equal; camlight headlight
```

Полученное изображение приведено на рис. 18.11.



Часть III. Математические публикации и компьютер

- Краткое введение в пакет LaTeX
- Редакторы и стандарты
- Интернет и математика

Результатом любого научного исследования являются публикации, подготовка и использование которых сегодня требует квалифицированного применения компьютера. Это касается редактирования текста, изготовления графических материалов, ведения библиографии, размещения электронных версий в Интернете, поиска статей и их просмотра. Де-факто сейчас стандартными системами подготовки научно-технических публикаций являются различные реализации пакета TeX и текстовый редактор Word. Кроме того, необходимы минимальные знания о стандартных форматах файлов, конверторах, программах и утилитах, используемых при подготовке публикаций.

Заключительная часть книги состоит из трех глав. В первой из них дается короткое введение в систему LaTeX, во второй рассматриваются различные редакторы и программы-конверторы, а заключительная глава посвящена Интернету. Понятно, что подробному описанию каждой из этих тем можно посвятить отдельную книгу, и мы здесь останавливаемся только на наиболее важных, на наш взгляд, и близких к теме книги вопросах.



Краткое введение в пакет LaTeX

Система верстки TeX была создана профессором Стэнфордского университета Дональдом Кнутом в 1977 году. Эта система представляет собой специализированный язык программирования [38], который включает команды, макроопределения и является аппаратно независимым, то есть работает одинаково на всех компьютерах, от PC до Cray. Позднее, в начале 80-х годов, Л. Лэмпорт [39] на основе TeX разработал издательскую систему LaTeX. Еще одно расширение, TeX – BibTeX, было написано О. Паташником и предназначено для подготовки списков литературы. В настоящее время TeX и LaTeX стали стандартом для научных публикаций в области естественных наук. В нашей книге мы рассматриваем одну из последних и наиболее распространенную версию – LaTeX2ε. Сейчас существует большое количество литературы по TeX, включая подробные руководства с детальным описанием всех возможностей (см. Список литературы). Мы ставим своей целью дать короткое описание, которое бы позволило быстро научиться создавать свои документы в LaTeX, используя наиболее распространенные команды, а для более глубокого изучения этого пакета советуем обращаться к книгам [30, 33].

Работа с LaTeX состоит из нескольких этапов, при этом пакет использует различные файлы, как исходные, так и те, которые создаются в процессе работы. Главным является файл с текстом документа (обычно с расширением .tex). Кроме того, в качестве входных используются файлы, содержащие информацию о структуре документа (расширение .cls), его разметке (.sty), библиографии (.bbl, .ind), шрифтов (.tfm, .fd) и др. Затем эти файлы обрабатываются транслятором, и результатом является файл с расширением .dvi (device independent – не зависящий от устройства). Файл .dvi содержит описание отформатированного текста, но в него не включены изображения шрифтов. Чтобы увидеть полученный текст, необходимо преобразовать этот файл с помощью соответствующего драйвера в нужный формат (PostScript, экран, принтер). Кроме того, LaTeX образует ряд вспомогательных файлов: протокол работы (с расширением .log или .lis), а также файлы с информацией о перекрестных ссылках (.aux), оглавлениях (.toc), списках иллюстраций (.lof), таблицах (.lot) и алфавитных указателях (.idx).

Издательская система на базе TeX состоит из следующих компонентов: транслятор языка TeX, dvi-драйверы, шрифты и тексты программ на языке TeX, которые определяют все команды для макропакетов (например, для LaTeX). Трансляторы и dvi-драйверы зависят от типа компьютера, а программы и шрифты являются аппаратно независимыми. TeX распространяется бесплатно, как и некоторые сервисные программы. Наиболее свежий и полный вариант всегда можно найти на сервере CTAN (Comprehensive TeX Archive Network), который размещен в Интернете по следующим адресам: ftp.shsu.edu, ftp.dante.de, ftp.tex.ac.uk.

В данной книге мы в основном опираемся на пакет MiKTeX, который вместе с текстовым редактором WinEdt представляет собой удобную среду для набора математических текстов на персональном компьютере с операционной системой Windows. Обе эти программы свободно распространяются через Интернет. Информация о том, где взять эти программы, как их инсталлировать и какие при этом могут возникнуть трудности, дана в главе 20 «Редакторы и стандарты». Теперь, после короткого введения, перейдем непосредственно к изложению команд и возможностей LaTeX2ε.

По умолчанию рабочим языком LaTeX является английский язык, для которого в стандарте пакета предусмотрены правила переноса, проверка грамматики и т. д. При помощи расширений LaTeX аналогичные возможности можно распространить и для большинства других языков. Вообще говоря, реализация языковой поддержки зависит от компьютера и операционной системы, так что для ее применения нужно посмотреть соответствующее описание. Для PC под Windows наиболее распространенным является пакет babel.

Начинающих пользователей пакета LaTeX иногда пугает множество терминов (команд) его языка и то, что при наборе, вообще говоря, не видно, каков будет результат. Однако для набора документа без особых изысков не требуется знания всех команд и можно обойтись их небольшим подмножеством. На самом деле работа с LaTeX проста, и лучше всего это показывает пример, который мы приводим ниже без комментариев. Если читатель обладает элементарными навыками в английском языке, то у него не должно возникнуть никаких проблем с пониманием этого примера. В тексте исходного документа слеш перед символом указывает на то, что это команда:

```
\documentclass[12pt]{article}
% Это строка комментария.
% Для подключения русского языка и кодировки Windows в
% пакете MikTeX используются следующие две команды
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\begin{document}
\title{Пример документа} % Заголовок документа
\author{Говорухин~В.~Н., Цибулин~В.~Г.} % Фамилии авторов
\date{1 января 2000 г.} % Дата формирования документа
\maketitle % Печатает заголовок
\begin{abstract}
```

Этот короткий документ является примером стандартного LaTeX-документа. В нем использованы часто употребимые команды. Аннотация выводится в выделенном абзаце уменьшенным шрифтом:

```
\end{abstract}
\section{Введение}
```

Обычный текст выводится стандартным для этого документа шрифтом. Размер шрифта определяется в заголовке документа (в этом документе 12 pt). Текст может разделяться на секции и главы. Эти разделы нумеруются автоматически. Здесь первым и единственным разделом является введение.

LaTeX позволяет легко набирать как простые формулы (например, $\sin{x^2}$), так и более сложные и нумеруемые автоматически:

```
% int - интеграл, frac - дробь
\begin{equation}
\int_{0}^\infty \frac{\sin x}{x} = \frac{\pi}{2}
\end{equation}
\begin{center}
\textbf{ \large{Подробнее о пакете говорится ниже.} }
\end{center}
\end{document}
```

На рис. 19.1 изображен результат, который получен после обработки этого файла компилятором и печати полученного dvi-файла (дан в уменьшенном масштабе).

Пример документа

Говорухин В. Н., Цибулин В. Г.

1 января 2000 г.

Аннотация

Этот короткий документ является примером стандартного LaTeX-документа. В нем использованы часто употребимые команды. Аннотация выводится в выделенном абзаце и уменьшенным шрифтом.

1 Введение

Обычный текст выводится стандартным для этого документа шрифтом. Размер шрифта определяется в заголовке документа (в этом документе 12pt). Текст может разделяться на секции и главы. Эти разделы нумеруются автоматически. Здесь первым и единственным разделом является введение. LaTeXпозволяет легко набирать как простые формулы (например, $\sin x^2$), так и более сложные и нумеруемые автоматически:

$$\int_0^\infty \frac{\sin x}{x} = \frac{\pi}{2} \qquad (1)$$

Подробнее о пакете говорится ниже.

Рис. 19.1. Результат обработки примера LaTeX-документа

Структура исходного файла и стили

Входным файлом для LaTeX является обычный текстовый файл, который, вообще говоря, может быть подготовлен в любом редакторе. В этом файле содержатся текст статьи, формулы, рисунки и ссылки на графические файлы, а также команды LaTeX, которые управляют параметрами отображения текста, размером страницы, характеристиками документа и т. д. Исходный текст не должен содержать разбивки на страницы и другого форматирования, а также переносов, которые программа расставит сама.

Символы и команды

Большинство символов в исходном текстовом файле будут отображаться естественным образом, то есть они обозначают сами себя, но существует ряд символов, которые используются в командах LaTeX и являются исключением из этого правила. Неправильное их применение может вызвать сообщение об ошибке. Это следующие символы:

```
$ & % # _ { } ~ ^ \.
```

Если интерпретатор встречает в строке символ «%», то все последующие символы этой строки им игнорируются в печатной версии, то есть этот символ можно применять для комментариев. Знак доллара ограничивает формулы, а фигурные скобки — группы в исходном файле. Знаки «^» и «_» применяются при наборе математических формул. Для того чтобы первые семь из этих символов вставить в документ, перед ними нужно поставить символ «\». Отметим, что система интерпретирует несколько стоящих подряд символов пробела как один, а стоящие в начале строки пробелы вообще игнорирует. Поясним сказанное примером. Во всех примерах в книге сначала дан LaTeX-текст, а затем — результат:

```
$x^2$ \$1=30rbl.  \# \#   \#
```

x^2 $1=30rbl. # # #

Список специальных математических и нематематических символов и соответствующие команды приведены в приложении.

Команды LaTeX начинаются с обратного слеша, после которого следует специальный символ или имя команды, состоящее только из латинских букв. Для некоторых команд после имени следуют параметры, заключенные в фигурные скобки. В именах команд прописные и строчные буквы различаются. Приведем простой демонстрационный пример:

```
\flushright Для \textit  {принудительного} перехода \\ на новую строку используется \\ два
\textbf{обратных} слеша. \\ Написано \today %Комментарий
```

Для *принудительного* перехода
на новую строку используется
два **обратных** слеша.
Написано 20 октября 2000 г.

Остановимся коротко на важном понятии — группе. Группой называется текст и команды, которые заключены в фигурные скобки. Действие команды распространяется только на стоящий в этих скобках текст.

Еще одна важная конструкция пакета — это окружение (environment). Под окружением понимается фрагмент исходного файла, заключенный между командами вида:

```
\begin{name of environment}
\end{name of environment}
```

Здесь name of environment представляет собой обязательный параметр и обозначает имя окружения. Например, окружение, определяющее нумерованные формулы, выглядит следующим образом:

```
\begin{equation} … \end{equation}
```

Многие команды пакета в качестве входных параметров требуют задания длин, например при установке нестандартных размеров страницы. В LaTeX предусмотрены сокращения для единиц длины, приведенные в табл. 19.1.

Таблица 19.1. Сокращения, принятые в LaTeX для единиц длины

Сокращение	Единица длины
mm	Миллиметр ≈ 1/25 дюйма
cm	Сантиметр = 100 миллиметров
in	Дюйм = 25.4 миллиметров
pt	Точка ≈ 1/72 дюйма ≈ 1/3 миллиметров
em	Приблизительная высота символа в текущем шрифте (m)
ex	Приблизительная ширина символа в текущем шрифте (x)

Структура исходного файла

Входной файл для LaTeX2ε должен начинаться следующей командой:

```
\documentclass[options]{class}
```

После этой команды могут стоять операторы, которые влияют на стиль всего документа или расширяют возможности пакета. Часть файла до следующей обязательной команды называется преамбулой, и в ней располагаются команды, определяющие характеристики всего документа. Следующей обязательной строкой файла является команда

```
\begin{document}
```

После этой команды следует текст документа со своими командами. Заканчиваться документ должен третьей обязательной командой:

```
\end{document}
```

Таким образом, минимальный LaTeX2ε файл имеет следующий вид:

```
\documentclass{article}
\begin{document}
```

```
Simple example
\end{document}
```

Иногда при наборе больших документов исходный файл удобно разделить на несколько файлов меньшего размера. Наиболее простую возможность сделать это предоставляет команда

```
\input{filename}
```

Она позволяет вставить содержимое файла с именем `filename` в любое место документа.

Преамбула документа

Теперь подробней остановимся на командах, которые могут стоять в преамбуле документа. Начнем с первой строки входного файла — команды

```
\documentclass[options]{class}
```

Эта команда специфицирует, какой документ должен быть в результате получен: параметр в фигурных скобках `class` определяет тип документа, а параметры в квадратных скобках (`options`) управляют его характеристиками. Существует четыре типа документов (`class`), которые могут быть присвоены по умолчанию (см. табл. 19.2).

Таблица 19.2. Стандартные типы документов в LaTeX

Тип	Описание
article	Статья для научных журналов, короткие доклады, письма и пр.
report	Документ, состоящий из нескольких глав (диссертация, небольшая книга)
book	Книга
slide	Слайды для проектора. В этом случае текст выводится большими буквами

Кроме стандартных могут применяться и типы, задаваемые пользователем, а также типы и стили различных издательств и редакций. Например, для того, чтобы использовать в документе тип статьи американского математического общества, документ должен начинаться командой

```
\documentclass{tran-l}
```

Параметры (`options`) команды `\documentclass` приведены в табл.19.3.

Базовый набор команд LaTeX не всегда достаточен для решения поставленных задач. Если требуется включение графических объектов в различных форматах, использование различных языков, цветного текста и т. п., то нужно использовать расширения стандартного набора, которые называются пакетами. Для активации пакета используется команда

```
\usepackage[options]{package}
```

Здесь `package` является именем пакета, а `options` его параметрами, которые, вообще говоря, не обязательны. Некоторые пакеты входят в состав базового сопровождения LaTeX, а многие распространяются отдельно. Приведем имена некоторых, наиболее распространенных пакетов в табл. 19.4, сопровождая их короткими описаниями.

Таблица 19.3. Параметры стандартных типов документов

Параметр	Описание
10pt, 11pt, 12pt	Размер основного шрифта документа (по умолчанию — 10 pt)
a4paper, letterpaper, b5paper, …	Размер бумаги для печати документа (по умолчанию — letterpaper)
Fleqn	Отображает формулы с выравниванием влево вместо центрирования
leqno	Размещает номера формул слева вместо размещения справа
landscape, portrait	Задает альбомную или портретную ориентацию бумаги (по умолчанию – portrait)
titlepage, notitlepage	Определяет, начинать ли текст после заголовка документа с новой страницы или нет
twocolumn	Вывод текста в две колонки
twoside, oneside	Задает выводную структуру документа для двусторонней или односторонней печати
openright, openany	Разрешает начинать главы только на нечетных страницах или на любых

Таблица 19.4. Пакеты LaTeX

Пакет	Описание
babel	Пакет языковой поддержки, в том числе и русификации
inputenc	Пакет задания кодировки текста документа (Win, KOI, ...)
graphicx	Пакет для импорта в документ рисунков в различных форматах
makeidx	Пакет для создания предметных и именных указателей
amssymb	Пакет для использования математических символов

Существует множество других пакетов. Например, чтобы в документе использовать текст, подготовленный пакетом Maple, в преамбуле документа должна стоять следующая команда:

```
\usepackage{maple2e}
```

Отметим, что при этом все файлы поддержки пакета (с расширением .sty) должны размещаться или в каталоге самого документа, или в каталоге LaTeX со стилевыми файлами.

Кроме того, в преамбуле документа могут находиться команды, управляющие параметрами страницы, команды, определяющие правила переноса слов, и др. Обо всем этом будет сказано далее.

Стили и параметры страницы

Результат обработки текста документа располагается в прямоугольной области на странице, которая называется телом документа. Над телом находится верхний колонтитул, а под ним — нижний колонтитул. Отступы слева и справа от тела текста называются полями и иногда используются под небольшие фрагменты текста — так называемые заметки на полях.

В LaTeX предусмотрено три стандартных стиля страниц, приведенные в табл. 19.5. Для их определения используются следующие две команды:

```
\pagestyle{style}
\thispagestyle{style}
```

Первая команда указывает на стиль страницы для всего документа, а вторая позволяет изменять стиль текущей страницы.

Таблица 19.5. Стандартные стили страницы

Стиль	Описание
plain	Стиль, используемый пакетом по умолчанию. Номер страницы печатается по центру нижнего колонтитула
headings	В верхнем колонтитуле печатается заголовок текущей главы и номер страницы
empty	Отсутствие колонтитулов и номеров страниц

По умолчанию для нумерации страниц используются арабские цифры. Стиль нумерации можно переопределить с помощью команды

```
\pagenumbering{num_style}
```

Стили num_style приведены в табл. 19.6.

Таблица 19.6. Стили нумерации страниц

Стиль	Вид нумерации страницы
Arabic	Арабские цифры
Roman или roman	Большие или маленькие римские цифры
Alph или alph	Большие или маленькие буквы

Геометрические параметры страницы также могут легко изменяться. Параметры страницы и соответствующие команды вместе со значениями, принятыми по умолчанию для листа размера letter, схематически представлены на рис. 19.2. Далее даны значения, принятые по умолчанию (они указаны на рисунке под соответствующими номерами):

1. one inch + \hoffset,
2. one inch + \voffset,
3. \oddsidemargin = 31 pt,
4. \topmargine = 28 pt,
5. \headheight = 12 pt,
6. \headsep = 20 pt,
7. \textheight = 526 pt,
8. \textwidth = 372 pt,
9. \marginparsep = 7 pt,
10. \marginparwidth = 71 pt,
11. \footskip = 36 pt,
12. \marginparpush = 5 pt,
13. \hoffset=0,
14. \voffset=0.

Последние три величины на рисунке не показаны.

Напомним, что размер бумаги указывается с помощью команды \documentclass. Иногда возникает необходимость указать другие параметры страницы для печати.

Это можно сделать, изменив значения величин, перечисленных на рис. 19.2. Для этих целей существуют две команды:

```
\setlength{parameter}{length}
\addtolength{parameter}{length}
```

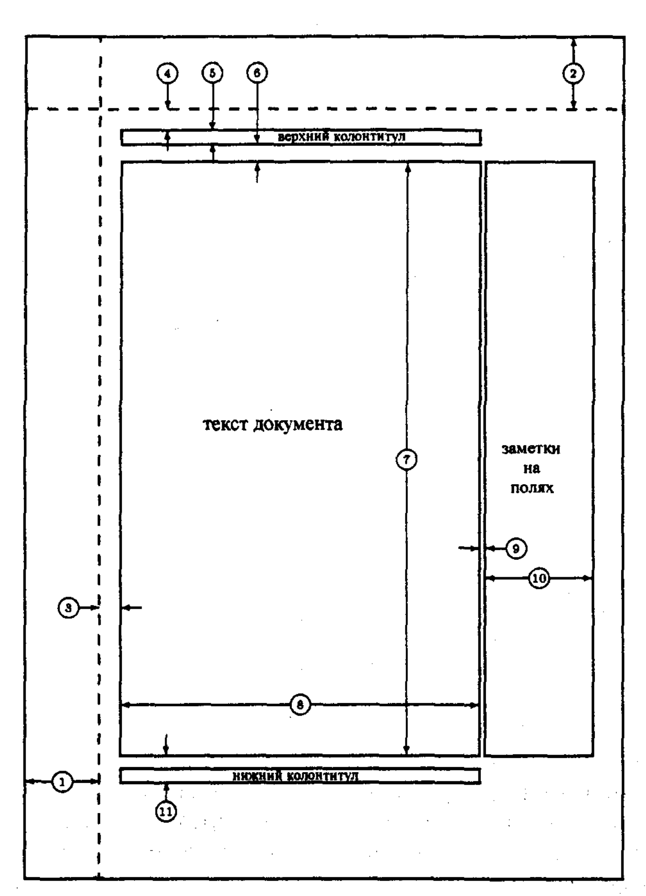


Рис. 19.2. Геометрические параметры страницы. Расшифровка обозначений дана в тексте

Первая команда присваивает величине parameter значение length, а вторая добавляет к величине parameter значение length. Например, для того чтобы увеличить ширину строки документа на 1 см, необходимо выполнить следующую команду:

```
\addtolength{\textwidth}{1cm}
```

Отметим, что команды, изменяющие параметры страницы, должны находиться в преамбуле документа.

Кроме описанных команд в преамбуле документа могут размещаться команды, определяемые пользователем (см. раздел этой главы «Программирование в LaTeX»), и некоторые другие команды. Приведем пример документа на русском языке в кодировке Windows в системе MiKTeX:

```
\documentclass[12pt,a4paper,russian]{article}
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\addtolength{\textwidth}{1cm}
\pagestyle{empty}
\begin{document}
Более сложный пример.
\end{document}
```

Набор текста

В предыдущем разделе рассматривалась структура документа и исходного файла, а также некоторые другие технические вопросы. Здесь мы остановимся на процессе набора текста документа, а подготовка математических выражений будет рассмотрена в разделе «Формулы» этой главы. Документ может включать в себя заголовок, аннотацию и разделяться на параграфы и главы. При наборе текста можно использовать различные символы, применять форматирование, управлять размерами символов, интервалами в тексте и т. п.

Заголовок документа

Документ обычно начинается с заголовка, который определяет группа команд, приведенных в табл. 19.7.

Таблица 19.7. Стандартные команды заголовка документа

Команда	Описание
\author{names}	Определяет имена авторов документа
\date{text}	Позволяет задать дату написания документа; если команда не используется, то LaTeX использует системную дату
\thanks{text}	Создает сноску внизу страницы
\title	Определяет название документа

Эти команды будут активированы, если после них стоит команда создания заголовка:

```
\maketitle
```

Если стиль документа `article`, то заголовок документа размещается перед текстом на первой странице, а для остальных стилей создается отдельная страница с заголовком.

Разделы, главы, абзацы, примечания

В LaTeX существует возможность разделять текст на разделы с различным уровнем вложенности при помощи команд, приведенных в табл. 19.8.

Таблица 19.8. Команды создания разделов документа

Команда	Описание
\part[opt]{text}	Самый высокий уровень вложенности. Не влияет на нумерацию глав
\chapter[opt]{text}	Доступно только для классов report и book. Перед текстом пользователя выводится слово Chapter (в русском варианте — глава) и номер главы
\section[opt]{text}	Номер и название раздела
\subsection[opt]{text}	Номер и название подраздела
\subsubsection[opt]{text}	Номер и название подраздела более низкого уровня вложенности
\paragraph[opt]{text}	Текст пользователя выделяется отступами и шрифтом
\subparagraph[opt]{text}	Самый низкий уровень вложенности
\appendix	Эта команда не выводит никакого текста, а служит для обнуления счетчиков

Во всех перечисленных командах параметр text содержит заголовок соответствующего раздела. Нумерация разделов по умолчанию происходит автоматически. Кроме разделения текста документа, эти команды используются при генерации оглавления документа и колонтитулов. В квадратных скобках может находиться последовательность символов, которая служит для замены текста заголовка в оглавлении.

Поясним сказанное примером:

```
\part{часть 1}
\section{Раздел 1}
\section[Прочитай это]{Раздел 2}
...
\appendix
\section{Приложение. Раздел 1}
```

В оглавлении вместо надписи Раздел 2 будет напечатано: Прочитай это.

Для определения сносок внизу страницы существует команда

```
\footnote[number]{text}
```

Здесь text — текст сноски, а number является необязательным параметром, предназначенным для изменения номера сноски. Отметим, что эту команду можно применять только в основном тексте документа.

Примечания на полях можно сделать при помощи команды

```
\marginpar{text}
```

По умолчанию текст выводится в правом поле страницы. Для того чтобы его выводить в левом поле, используется команда \reversemarginpar, а для возвращения значения по умолчанию надо выполнить команду \normalmarginpar.

Если во входном файле встречается пустая строка, то следующий за ней текст будет выводиться начиная с нового абзаца. По умолчанию пакет выравнивает абзацы по ширине. Несколько пустых строк считаются за одну (аналогично правилу для пробелов), а для изменения интервалов существуют специальные команды. Перейти к новому абзацу можно и при помощи команды

\par

Отметим, что пустая строка не должна встречаться в тех местах файла, где невозможно начать текст с нового абзаца, например в математических формулах это приведет к ошибке при трансляции.

По умолчанию первая строка для первого в разделе абзаца печатается без отступа, а первые строки для последующих абзацев — с отступом. Однако эти правила можно изменить при помощи команд, приведенных в табл. 19.9.

Таблица 19.9. Команды управления абзацным отступом

Команда	Описание
\indent	Эта команда производит горизонтальный отступ в начале абзаца
\noindent	Эта команда подавляет горизонтальный отступ в начале абзаца

Разрывы, интервалы, переносы

При создании печатного варианта LaTeX автоматически разбивает документ на страницы, сам выравнивает строки и делает переносы. Но часто возникают ситуации, когда необходимо это сделать принудительно, если автоматический вариант не удовлетворяет нашим пожеланиям. Для управления разрывами существуют команды, приведенные в табл. 19.10.

Таблица 19.10. Команды управления разрывами абзацев и строк

Команда	Описание
\\ или \newline	Конец строки и переход на новый абзац
*	Конец строки без перехода на новый абзац
\newpage	Переход на новую страницу
\pagebreak[number]	Разрыв страницы
\nopagebreak[number]	Запрет перехода на новую страницу
\linebreak[number]	Переход на новую строку
\nolinebreak[number]	Запрет перехода на новую строку

Необязательный параметр number служит для превращения команды из требования в пожелание. Величина number должна быть между 0 и 4 и управляет степенью обязательности применения команды: значение 4 соответствует наибольшей обязательности, а 0 позволяет LaTeX игнорировать команду, если ее применение приводит к плохому результату.

Иногда может понадобиться изменить размер поля текста одной из страниц, обычно на небольшую величину. Для этого служит команда

```
\enlargethispage{size}
```

При выравнивании границы текста пакет сам изменяет интервал между словами. В конце предложения обычно интервал больше, что делает текст более наглядным. Однако можно управлять и этим. Для запрета автоматического изменения интервала служит символ «~». Например, инициалы не будут отрываться от фамилии, если их писать так:

```
Говорухин~В.~Н.
```

Теперь опишем, как изменять интервалы. Для задания межстрочного интервала используется команда

```
\linespread{factor}
```

Эта команда должна находиться в преамбуле документа. Здесь factor определяет размер интервала, \linespread{1.3} соответствует полуторному интервалу, а \linespread{1.6} — двойному. По умолчанию factor=1, что соответствует одинарному интервалу.

Также в преамбуле могут стоять и следующие команды:

```
\setlength{\parindent}{value}
\setlength{\parskip}{value}
```

Первая задает расстояние value отступа в первой строке абзаца, а вторая определяет интервал между абзацами.

Расстояние между словами в строке пакет определяет автоматически, а для принудительного изменения интервала используется команда

```
\hspace{length}
```

Расстояние между параграфами, главами и прочим определяется системой LaTeX автоматически для всего документа. Однако, если это необходимо, можно увеличить вертикальный интервал в нужном месте, указав его явно:

```
\vspace{length} или \\[length]
```

В последних трех командах length является длиной отступа в любых единицах, разрешенных в LaTeX (см. раздел «Символы и команды» этой главы).

Пример:

```
\noindent Первый абзац без отступа слева. \\ Переход на новую строку.
\hspace{0.3cm} Второй абзац с дополнительным отступом слева.
\vspace{0.1in} Третий абзац с дополнительным отступом сверху.
```

Первый абзац без отступа слева.
Переход на новую строку.

Второй абзац с дополнительным отступом слева.

Третий абзац с дополнительным отступом сверху.

Расстановкой переносов и плотностью строки управляют команды \fussy и \sloppy. Первая команда активна по умолчанию и приводит к тому, что транслятор старается оставить как можно меньше лишних пробелов в строке. Однако это может привести к тому, что строка вылезет за правый край страницы. Ослабляет действие такого режима вторая команда, которая позволяет установить режим без выравнивания и без переносов. Она имеет вид:

```
\raggedright
```

Не всегда алгоритм расстановки переносов работает хорошо, бывают ситуации, когда символы выходят за поле текста. В этом случае пользователь может дополнить базовые правила переносов при помощи следующей команды:

```
\hyphenation{word list}
```

Эта команда должна находиться в преамбуле документа. В скобках даются слова, для которых определяются правила переносов, причем места возможных переносов помечаются символом «-», а слова отделяются друг от друга пробелами. Например:

```
\hyphenation{па-ра-маг-не-тик син-гу-ляр-ность}
```

Существует еще возможность жесткого принудительного переноса командой

```
\-
```

Пакет всегда будет делать перенос на месте, помеченном этими символами, конечно, если слово стоит в конце строки.

Часто возникает и обратная ситуация, когда слово или фразу нельзя разрывать. Для запрещения переноса служит команда

```
\mbox{text}
```

Приведем пример:

```
Поставим принудительный перенос с оши\-бкой. \mbox{Теперь попросим эту фразу
не разрывать.} Новый абзац.  \noindent А это абзац номер~2 без отступа.
```

Поставим принудительный перенос с оши-
бкой. Теперь попросим эту фразу не разрывать.
    Новый абзац.
А это абзац номер 2 без отступа.

Шрифты, размеры, специальные и национальные символы

LaTeX сам выбирает подходящие шрифты исходя из стиля документа. В некоторых случаях возникает необходимость изменения шрифта или его размера вручную. Отметим, что реальный размер символов будет зависеть от установленного базового размера, то есть от стиля (класса) документа, а все остальные шрифты изменяются пропорционально ему. Для изменения шрифта текста, стоящего в фигурных скобках, имеются команды, приведенные в табл.19.10.

Таблица 19.11. Команды, определяющие шрифт текста

\textrm{...}	roman	\textsf{...}	sans serif
\texttt{...}	typewriter		
\textmd{...}	medium	\textbf{...}	**bold face**
\textup{...}	upright	\textit{...}	*italic*
\textsl{...}	*slanted*	\textsc{...}	SMALL CAPS
\emph{...}	*emphasised*	\textnormal{...}	document font

В табл. 19.11 в каждой из двух колонок слева дана команда, а справа — название шрифта, выведенное с использованием этого шрифта.

Другим способом задания шрифта является конструкция вида

```
\begin{family} … \end{family}
```

Переменная family обозначает одно из семейств шрифтов. Список семейств приведен в таблице 19.12; в каждой колонке имя семейства находится слева, а справа — тип выводимого текста.

Таблица 19.12. Названия семейств шрифтов

rmfamily - \textrm	itshape - \textit
mdseries - \textmd	bfseries - \textbf
upshape - \textup	slshape - \textsl
sffamily - \textsf	scshape - \textsc
ttfamily - \texttt	normalfont - \textnormal

Кроме того, переменные family могут использоваться для определения типа шрифта как команды без параметра, например, чтобы установить наклонный шрифт, можно выполнить команду

```
\itshape
```

Размеры символов изменяются после применения команд, которые даны в табл. 19.13.

Таблица 19.13. Команды управления размером символа

\tiny	tiny font	\Large	larger font
\scriptsize	very small font	\LARGE	very large font
\footnotesize	quite small font	\huge	huge
\small	small font	\Huge	largest
\normalsize	normal font		
\large	large font		

После того как транслятор встретит одну из перечисленных команд, все символы в группе будут выводиться этим размером. Напомним, что группой называются команды и символы, объединенные фигурными скобками. Значит, для того, чтобы только часть текста выводилась определенным шрифтом, достаточно этот текст заключить в фигурные скобки.

Проиллюстрируем сказанное примером:

```
\itshape Этот текст \textbf{иллюстрирует} {\large различные способы изменения шрифта}.
\normalfont Изменяются {\small как} типы \begin{scshape} шрифтов. \end{scshape} \Large так
и \LARGE \textsf{размер} \Huge символов.
```

Этот текст ***иллюстрирует***

различные способы изменения

шрифта. Изменяются как типы

шрифтов, так и размер

символов.

Иногда требуется выделить отдельные фразы или слова другим шрифтом. Для выделения текста курсивом на фоне прямого шрифта и прямым шрифтом на фоне курсива служит команда

```
\emph{text}
```

Эта команда имеет обязательный параметр — текст. Другим способом выделения является подчеркивание и заключение части текста в рамку:

```
\underline{text}
\fbox{text}
```

Пример такого выделения:

```
Выделение \emph{курсивом} на фоне прямого текста. \itshape Выделение \emph{прямым} шрифтом.
\\* \underline{Подчеркнем фразу}. А слово \fbox{рамка} заключим в рамку.
```

Выделение *курсивом* на фоне прямого текста. *Выделение* прямым *шрифтом.*
Подчеркнем фразу. А слово рамка *заключим в рамку.*

Как уже отмечалось, большинство символов набираются естественно: точка в исходном файле преобразуется в точку при печати. Однако бывают нужны символы, которые требуют специального набора. Двойные кавычки в LaTeX задаются двумя одинарными: "text". Существует возможность воспроизводить тире различной длины. Это осуществляется через повторение символа «–» (минус), причем длина тире зависит от количества минусов. Для набора многоточия используется команда ldots. Знак параграфа набирается с помощью символа «\S», а знак «©» — командой copyright. Наконец, любой символ можно набрать, зная его код, при помощи команды symbol{code}. Кроме того, в тексте можно использовать и математические символы, речь о которых пойдет в разделе «Формулы» этой главы. Список в табл. 19.14 показывает, как получать буквы и символы национальных алфавитов. В каждой из четырех колонок символ дан слева, а справа — соответствующая команда.

Таблица 19.14. Символы национальных алфавитов

ò	\\'o	ó	\\'o	ô	\^o	õ	\~o
ō	\=o	ȯ	\\.o	ö	\\"o	ç	\c c
ŏ	\u o	ǒ	\v o	ő	\H o	ǫ	\c o
ọ	\d o	o̲	\b o	o͡o	\t oo		
œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
å	\aa	Å	\AA				
ø	\o	Ø	\O	ł	\l	Ł	\L
ı	\i	ȷ	\j	¡	!‘	¿	?‘

Часто в полиграфическом оформлении используются линейки, как вертикальные, так и горизонтальные. Для их создания применяются команды:

```
\rule{opt1}{opt2}          \hrule          \vrule
```

Первая команда имеет два обязательных параметра opt1 и opt2, которые описывают соответственно ширину и высоту линии. Две другие команды проводят соответственно горизонтальную и вертикальную линии стандартной толщины.

Покажем пример использования описанных команд:

```
Урок по-французски пишется Le{\c c}on, а гостиница - H\^otel. \rule{0.5cm}{0.5cm} Многоточие
(\ldots) задается специальной командой, а не тремя точками. Знаку ""параграфа" (\S)
соответствует команда слеш + S. \hrule \smallskip \copyright \symbol{190} Авторы\symbol{191}
```

Урок по-французски пишется Leçon, а гостиница — Hôtel. ■ Многоточие (...) задается специальной командой, а не тремя точками. Знаку "параграфа" (§) соответствует команда слеш + S.
©«Авторы»

Теперь несколько слов о поддержке языков, отличных от английского, в частности о русификации. Вообще говоря, подключение языковой поддержки зависит от конкретной реализации пакета. Наиболее распространенным в настоящее время является пакет babel, который должен подключаться при помощи команды `\usepackage[language]{babel}`. Кроме того, для некоторых языков, в частности, для русского, существует несколько вариантов кодировок символов (КОИ-8, Windows, ...). В этом случае кодировка должна быть задана при помощи подключения пакета inputenc. Пример использования этих пакетов применительно к русскому языку и кодировке Windows дан в разделе, посвященном преамбуле документа.

Формат и типы абзацев, блоки

По умолчанию документ будет печататься в одну колонку. Если необходимо печатать в две колонки весь документ, то надо указать параметр `twocolumn` в команде `\documentclass`. Если же выводить в две колонки надо только часть текста, то используется команда `twocolumn` внутри документа. После этой команды с новой страницы текст будет выводиться в две колонки. Кроме того, существуют и другие возможности вывода текста в несколько колонок (таблицы, блоки).

В предыдущих разделах уже рассматривался вопрос об абзацах, интервалах между ними и пр. Напомним, что по умолчанию пакет LaTeX выравнивает абзацы по ширине. Однако существуют и другие возможности, почти все они реализованы в виде окружений.

Для того чтобы набрать часть текста с отступом от краев (важная фраза, цитата и др.), надо использовать окружение `quote`. Существуют еще два похожих окружения — `quotation` и `verse`. Приведем пример выделения текста:

```
Для выделения важных мест текста можно использовать специальное окружение. \begin{quote}
Текст внутри окружения выводится с дополнительными отступами.\end{quote} Часто такое
выделение используется для аннотаций.
```

Для выделения важных мест текста можно использовать специальное окружение.

> Текст внутри окружения выводится с дополнительными отступами.

Часто такое выделение используется для аннотаций.

Если текст заключен между командами \begin{verbatim} и \end{verbatim}, то никакие команды LaTeX в этом тексте не будут восприниматься, и он напечатается текстом пишущей машинки, так же как и в исходном файле. Например:

```
\begin{verbatim}Пример команды LaTeX: \ldots \end{verbatim}
```

```
Пример команды LaTeX:
\ldots
```

Тексты с выравниванием по центру, левому или правому краю создаются при помощи окружений flushleft, flushright и center соответственно. Например:

```
\begin{flushleft} Пример абзаца, выровненного по левому краю. Пакет не пытается получить
строки одинаковой длины. \end{flushleft} \begin{flushright} Пример абзаца, выровненного по
правому краю. При этом с длиной строк \LaTeX{} поступает аналогично предыдущему случаю.
\end{flushright} \begin{center} В случае центрированного текста \\ все строки \\
центрируются \end{center}
```

Пример абзаца, выровненного по левому
краю. Пакет не пытается получить строки
одинаковой длины.

Пример абзаца, выровненного по правому
краю. При этом с длиной строк LaTeX
поступает аналогично предыдущему случаю.

В случае центрированного текста
все строки
центрируются

Для создания простых перечней (списков) применяется окружение itemize. Если нужен нумерованный список, то следует прибегнуть к окружению enumerate. Если же создается перечень, где каждый элемент имеет свой заголовок, то используется окружение description. Во всех трех случаях элементы перечня задаются командой item, которая в зависимости от целей может иметь параметры. Отметим, что до первой команды в перечнях могут стоять только команды управления шрифтами и подобные им, но не должно быть никакого текста. Перечни могут быть вложенными друг в друга. Максимальная глубина вложенности равна четырем. Примеры перечней приведены далее:

```
Математическое исследование: \begin{itemize} \item Поиск литературы. \item Выбор методов
решения.\end{itemize}
```

Математическое исследование:

- Поиск литературы.
- Выбор методов решения.

```
Использование компьютера: \begin{enumerate} \item Символьные вычисления. \item Численный
анализ. \item Публикации. \end{enumerate}
```

Использование компьютера:

1. Символьные вычисления.
2. Численный анализ.
3. Публикации.

```
\begin{description} \item[Maple] - пакет для символьных вычислений. \item[Matlab] – стандарт для численного анализа \end{description}
```

Maple - пакет для символьных вычислений.

Matlab – стандарт для численного анализа

Для создания нестандартных абзацев, например с отступом всех строк, за исключением первой, можно использовать две команды:

\hangindent и \hangafter

Первая из них указывает величину отступа строк от полей, а вторая — номер строки, после которой этот отступ делать. Отметим, что эти параметры могут принимать и отрицательные значения. Например:

```
\noindent \hangindent=1cm \hangafter=2 Пример абзаца, в котором отступ на один сантиметр начинается с третьей строки.
```

Пример абзаца, в котором отступ
на один сантиметр начинается с тре-
    тьей строки.

Основным объектом, с которым работает транслятор пакета, является блок, в качестве которого может фигурировать как одна буква, так и целые фразы. Выше была рассмотрена команда \mbox, которая делает неразрывной целую фразу, точнее делает ее как бы одним словом. Существует команда, в которой можно указывать ширину блока:

\makebox[length][par]{text}

Здесь length указывает на длину строки в блоке, причем длина может задаваться как единицами размера, так и при помощи слова или последовательности символов, text — текст, находящийся в блоке. В качестве необязательного параметра par могут выступать символы r, l, c. Параметр l означает выравнивание влево, r — выравнивание вправо, а c — выравнивание по центру. В частности, эта команда позволяет накладывать один текст на другой:

```
тест \makebox[6ex]{123} тест. \makebox[0pt][l] {/////////} Наложение
```

тест   123   тест. ~~Наложение~~

Для создания блока переменной высоты используется команда:

\parbox[pos]{length}{text}

Необязательный параметр pos определяет вертикальное позиционирование блока и может принимать следующие значения: t — верхняя строка блока выравнивается по текущей строке, c — середина блока выравнивается по середине текущей строки, b — нижняя строка блока выравнивается по текущей строке. Выше говорилось, что для заключения в рамку фрагмента текста применяется команда \fbox. Существует также команда \framebox для взятия текста в рамку, назначение параметров которой совпадает с описанным для команды \makebox. И наконец, для сдвига блока вверх относительно линии строки имеется команда с параметрами, аналогичными команде \parbox:

\raisebox{height}{text}

Приведем пример:

```
Вставим \parbox{2.5cm}{абзац в строку, которая} продолжается. Теперь \framebox[1in][r]{слово} поместим в рамку, а затем поднимем точки вверх \raisebox{3pt}{\ldots}
```

Вставим абзац в строку, которая продолжается. Теперь слово поместим в рамку, а затем поднимем точки вверх ···

Описанные блочные команды используются обычно для небольших фрагментов текста. Для создания больших блоков, включающих различные объекты (таблицы или рисунки), существует окружение minipage, параметры которого аналогичны параметрам окружения parbox:

```
\begin{minipage}[pos]{width}
...
\end{minipage}
```

Ссылки и нумерация

LaTeX позволяет организовать ссылки на страницы документа или его разделы так, чтобы транслятор автоматически определял номер страницы для ссылки. Для этого в месте, на которое нужно ссылаться, ставится метка при помощи команды

```
\label{name}
```

Здесь name служит именем, на которое в файле можно ссылаться одним из следующих способов:

```
\pageref{name} или \ref{name}
```

Первая из этих команд ссылается на номер страницы, на которой стоит метка, а вторая — на номер главы, секции, формулы или другого нумерованного объекта.

Если в файле есть ссылки, то для правильной его обработки транслятор необходимо запускать два раза. После первого прохода LaTeX записывает информацию о ссылках в специальный файл, а при втором запуске считывает эту информацию и расставляет ссылки. Если после второго прохода будут появляться сообщения об ошибках (после первого запуска не обращайте на них внимания), то ошибки действительно есть и их надо искать.

Многие объекты пакет нумерует автоматически. Например, каждой главе, секции или отдельной формуле соответствует номер. Эта операция осуществляется при помощи стандартных счетчиков, а также счетчиков, создаваемых пользователем (о них речь пойдет в разделе «Программирование в LaTeX» этой главы). Имя счетчика обычно совпадает с именем использующего его окружения или команды. В табл. 19.15 приведены имена основных стандартных счетчиков.

Таблица 19.15. Имена стандартных счетчиков

part	**paragraph**	**figure**
chapter	**subparagraph**	**table**
section	**page**	**footnote**
subsection	**equation**	**mpfootnote**
subsubsection		

Чтобы вывести номер счетчика, используется команда

```
\thename
```

Здесь name является именем счетчика. Например, чтобы вывести номер страницы, нужно выполнить команду \thepage. Значение счетчика обычно дается положительным целым числом. В зависимости от определения команды задаваемое счетчиком число может выводиться арабскими, римскими цифрами или специальными символами. Команды, работающие со счетчиками, сначала изменяют их значение, а потом используют. Большинство счетчиков при начале работы равны нулю. Для изменения значения счетчика есть две команды:

```
\setcounter{name}{num}
\addtocounter{name}{num}
```

Первая команда устанавливает счетчику с именем name значение num, а вторая увеличивает значение счетчика name на величину num.

Пример:

```
\subsection{Раздел 1} \label{lbl} Этот раздел помечен меткой \textit{lbl}. Теперь увеличим
значение счетчика разделов на 5. \addtocounter{subsection}{5} \subsection{Раздел 2}Мы
изменили нумерацию на странице \thepage\ в разделе \ref{lbl}.
```

0.1 Раздел 1

Этот раздел помечен меткой *lb1*. Теперь увеличим значение счетчика разделов на 5.

0.7 Раздел 2

Мы изменили нумерацию на странице 1 в разделе 0.1.

Некоторые команды и окружения существуют в двух вариантах: второй отличается от основного тем, что после имени следует символ «*», который сообщает пакету о том, что данный объект не надо нумеровать.

Формулы

В этом разделе мы рассмотрим одно из главных назначений системы TeX – набор формул. Новичков часто пугает то, что при наборе формул в LaTeX они не отражаются моментально на экране. Это вызывает трудности только в самом начале освоения пакета. На самом деле запись формул проста: математические выражения записываются так же, как при их чтении вслух. Например, чтобы записать формулу $y=x^2$, достаточно набрать: $y=x^2$.

Далее мы остановимся на возможностях базового комплекта пакета LaTeX2ε. Если читатель не найдет подходящего решения при помощи этого набора, то это означает, что пора обратиться к расширениям LaTeX, например к пакету AMS, который был создан под эгидой Американского математического общества.

Для набора математических выражений пакет имеет специальные режимы, которые реализованы в виде окружений и команд. Различают математические формулы внутри текста и так называемые «выключные», то есть выделенные в отдельную строку.

По умолчанию все выключные формулы выравниваются по центру строки, а изменить этот режим можно при помощи параметра fleqn команды \documentclass.

Для формул внутри текста используется команда, имеющая три варианта написания:

```
\begin{math} … \end{math}
\( … \)
$ … $
```

Вместо троеточия должны стоять команды, задающие формулы. Хотя все три варианта равноправны, для коротких формул в основном используется последний.

Для создания ненумерованных выделенных в строку уравнений используются следующие конструкции:

```
\begin{displaymath} … \end{displaymath}
\[ … \]
```

Существует еще и короткий вариант, когда команды заключаются между символами $$ … $$, но в этом случае будет игнорироваться изменение режима центрирования формул.

Создание выделенных уравнений с их автоматической нумерацией происходит при помощи команды

```
\begin{equation} … \end{equation}
```

Это уравнение можно пометить командой \label и ссылаться на него в любом месте документа, причем ссылки будут расставляться автоматически. При нумерации всех формул используется один счетчик, а формат счетчика зависит от класса документа. Для класса article происходит сквозная нумерация, а для book или report нумерация ведется по главам.

Автоматической нумерацией формул можно не пользоваться, а расставлять номера вручную. Для этого применяются команды \eqno или \leqno (в первом случае номер формулы будет стоять справа, а во втором — слева).

Приведем простой пример:

```
Эта формула $r=\sqrt{x^2+y^2}$ стоит внутри текста. Теперь поместим выделенную формулу без
нумерации \[ E=m c^2.\] а после нее - с нумерацией: \begin{equation} \label{equat} \sin^2 x
+ \cos^2 x = 1. \end{equation}  Формула (\ref{equat}) должна быть известна всем. Поставим в
следующей формуле номер слева: $$ 7 \times 8 = 56 \leqno(**) $$
```

Эта формула $r = \sqrt{x^2 + y^2}$ стоит внутри текста. Теперь поместим выделенную формулу без нумерации

$$E = mc^2,$$

а после нее – с нумерацией:

$$\sin^2 x + \cos^2 x = 1. \qquad (1)$$

Формула (1) должна быть известна всем. Поставим в следующей формуле номер слева:

$$(**) \qquad 7 \times 8 = 56$$

Нумерация последней формулы показывает, что в качестве номера не обязательно должны быть цифры.

Описанные команды предназначены для набора простых формул, помещающихся в основном на одной строке. Для набора матриц и систем уравнений используются окружения array и eqnarray, о которых будет говориться далее.

Отметим, что при печати написание формул, выделенных в строку, и формул, стоящих в тексте, может отличаться, например:

```
Формула в тексте $\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$.
Выделенная в строку формула: \begin{displaymath} \lim_{n \to \infty} \sum_{k=1}^n
\frac{1}{k^2}=\frac{\pi^2}{6} \end{displaymath}
```

Формула в тексте $\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$.
Выделенная в строку формула:

$$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$$

Если в тексте встречается формула, то LaTeX переходит в так называемую математическую моду. При этом анализ выражения происходит по особым правилам с применением специальных команд.

Символы и шрифты в формулах

В LaTeX можно набрать практически все мыслимые символы, начиная от знаков математических операций до букв древнееврейского алфавита. Большинство этих символов отсутствует на клавиатуре компьютера, и для их набора надо знать соответствующие команды. Их перечень дан ниже в этом разделе. Отметим, что большинство этих команд работает только в математической моде, причем для использования некоторых команд нужно подключить соответствующие пакеты.

Как уже отмечалось, несколько пробелов подряд пакет считает за один. Однако возникают ситуации, особенно при наборе формул, когда нужно поставить большой или, наоборот, маленький интервал между символами. В табл. 19.16 перечислены команды для установки различных пробелов.

Таблица 19.16. Команды установки интервалов в математической моде

\,	узкий	\:	средний
\;	широкий	\!	отрицательный
\quad	очень широкий	\qquad	самый широкий

Пример:

```
$a\,b\:c\;d\!e\quad f \qquad g$
```

$a\,b\:c\;d\!e\quad f \qquad g$

Для пунктуации в формулах используются многоточия, запятые и другие символы, перечень которых дан в табл. 19.17.

Таблица 19.17. Знаки пунктуации и многоточия

$,$	,	$;$	;	$:$	\colon
$.$	\ldotp	$\cdot$	\cdotp	$\ldots$	\ldots
$\cdots$	\cdots	$\vdots$	\vdots	$\ddots$	\ddots

Подчеркнем, что команды акцентирования в математической моде отличаются от команд в текстовой моде. Например, в математической моде символ «'» используется для обозначения производной или штриха. Список команд акцентирования приведен в табл. 19.16.

Таблица 19.18. Команды акцентирования в математической моде

$\hat{a}$	\hat{a}	$\check{a}$	\check{a}	$\tilde{a}$	\tilde{a}	$\acute{a}$	\acute{a}
$\grave{a}$	\grave{a}	$\dot{a}$	\dot{a}	$\ddot{a}$	\ddot{a}	$\breve{a}$	\breve{a}
$\bar{a}$	\bar{a}	$\vec{a}$	\vec{a}	$\widehat{A}$	\widehat{A}	$\widetilde{A}$	\widetilde{A}

Обращение к этим командам происходит стандартно:

```
$\vec a, \quad \dot{c}, \qquad \widehat{DE}$
```

$\vec a, \quad \dot{c}, \qquad \widehat{DE}$

При отсутствии нужного акцента необходимую комбинацию можно подготовить при помощи команды \stackrel{верх}{низ}. Чтобы получить в математической формуле изображение перечеркнутого символа, надо перед соответствующей командой поставить команду not:

```
$v \stackrel{\mathrm{def}}{\equiv} 0, \quad \{ x : x \not\in X \}$
```

$v \stackrel{\mathrm{def}}{\equiv} 0, \quad \{ x : x \not\in X \}$

Для проведения линии над выражением и подчеркивания выражения существуют соответственно команды:

\overline и \underline

Чтобы провести над выражением левую или правую стрелку, надо обратиться соответственно к командам \overleftarrow и \overrightarrow

Пример:

```
\begin{displaymath} \overrightarrow{ABCD} \qquad \underline{l+m+n} \end{displaymath}
```

$$\overrightarrow{ABCD} \qquad \underline{l+m+n}$$

Далее, в таблицах 19.19–19.25 приведены команды из стандартного набора LaTeX, которые реализуют символы, использующиеся в математических выражениях.

Для получения символов отрицания каждой операции из табл. 19.19 используется приставка not перед именем команды.

Символы из табл. 19.25 могут использоваться и в текстовом режиме. Теперь приведем команды пакета AMS, реализующие специальные математические символы (табл. 19.26–19.31). Напомним, что для вызова этих команд необходимо предварительно подключить соответствующий пакет в преамбуле документа. Если пакеты не установлены, то их можно найти в Интернете на сервере CTAN.

Таблица 19.19. Символы логических отношений

$<$	<	$>$	>	$=$	=
$\leq$	\leq or \le	$\geq$	\geq or \ge	$\equiv$	\equiv
$\ll$	\ll	$\gg$	\gg	$\doteq$	\doteq
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong
$\sqsubset$	\sqsubset [a]	$\sqsupset$	\sqsupset [a]	$\Join$	\Join [a]
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\bowtie$	\bowtie
$\in$	\in	$\ni$	\ni , \owns	$\propto$	\propto
$\vdash$	\vdash	$\dashv$	\dashv	$\models$	\models
$\mid$	\mid	$\parallel$	\parallel	$\perp$	\perp
$\smile$	\smile	$\frown$	\frown	$\asymp$	\asymp
$:$	:	$\notin$	\notin	$\neq$	\neq or \ne

Таблица 19.20. Маленькие греческие буквы

α	\alpha	θ	\theta	o	o	υ	\upsilon
β	\beta	ϑ	\vartheta	π	\pi	ϕ	\phi
γ	\gamma	ι	\iota	ϖ	\varpi	φ	\varphi
δ	\delta	κ	\kappa	ρ	\rho	χ	\chi
ϵ	\epsilon	λ	\lambda	ϱ	\varrho	ψ	\psi
ε	\varepsilon	μ	\mu	σ	\sigma	ω	\omega
ζ	\zeta	ν	\nu	ς	\varsigma		
η	\eta	ξ	\xi	τ	\tau		

Таблица 19.21. Большие греческие буквы

Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

Таблица 19.22. Символы логических операций

$+$	+	$-$	-		
$\pm$	\pm	$\mp$	\mp	$\triangleleft$	\triangleleft
$\cdot$	\cdot	$\div$	\div	$\triangleright$	\triangleright
$\times$	\times	$\setminus$	\setminus	$\star$	\star
$\cup$	\cup	$\cap$	\cap	$\ast$	\ast
$\sqcup$	\sqcup	$\sqcap$	\sqcap	$\circ$	\circ
$\vee$	\vee , \lor	$\wedge$	\wedge , \land	$\bullet$	\bullet
$\oplus$	\oplus	$\ominus$	\ominus	$\diamond$	\diamond
$\odot$	\odot	$\oslash$	\oslash	$\uplus$	\uplus
$\otimes$	\otimes	$\bigcirc$	\bigcirc	$\amalg$	\amalg
$\bigtriangleup$	\bigtriangleup	$\bigtriangledown$	\bigtriangledown	$\dagger$	\dagger
$\lhd$	\lhd [a]	$\rhd$	\rhd [a]	$\ddagger$	\ddagger
$\unlhd$	\unlhd [a]	$\unrhd$	\unrhd [a]	$\wr$	\wr

Таблица 19.23. Стрелки

$\leftarrow$	\leftarrow or \gets	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\rightarrow$	\rightarrow or \to	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow
$\rightleftharpoons$	\rightleftharpoons	$\iff$	\iff (bigger spaces)	$\leadsto$	\leadsto [a]

Таблица 19.24. Сопутствующие символы

$\dots$	\dots	$\cdots$	\cdots	$\vdots$	\vdots	$\ddots$	\ddots
$\hbar$	\hbar	$\imath$	\imath	$\jmath$	\jmath	ℓ	\ell
$\Re$	\Re	$\Im$	\Im	$\aleph$	\aleph	$\wp$	\wp
$\forall$	\forall	$\exists$	\exists	$\mho$	\mho [a]	∂	\partial
$'$	'	$\prime$	\prime	$\emptyset$	\emptyset	∞	\infty
∇	\nabla	$\triangle$	\triangle	$\Box$	\Box [a]	$\Diamond$	\Diamond [a]
$\bot$	\bot	$\top$	\top	$\angle$	\angle	$\surd$	\surd
$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit	$\clubsuit$	\clubsuit	$\spadesuit$	\spadesuit
$\neg$	\neg or \lnot	$\flat$	\flat	$\natural$	\natural	$\sharp$	\sharp

Таблица 19.25. Нематематические символы

†	\dag	§	\S	©	\copyright
‡	\ddag	¶	\P	£	\pounds

Таблица 19.26. Стрелки из пакета AMS

$\dashleftarrow$	\dashleftarrow	$\dashrightarrow$	\dashrightarrow	$\multimap$	\multimap
$\leftleftarrows$	\leftleftarrows	$\rightrightarrows$	\rightrightarrows	$\upuparrows$	\upuparrows
$\leftrightarrows$	\leftrightarrows	$\rightleftarrows$	\rightleftarrows	$\downdownarrows$	\downdownarrows
$\Lleftarrow$	\Lleftarrow	$\Rrightarrow$	\Rrightarrow	$\upharpoonleft$	\upharpoonleft
$\twoheadleftarrow$	\twoheadleftarrow	$\twoheadrightarrow$	\twoheadrightarrow	$\upharpoonright$	\upharpoonright
$\leftarrowtail$	\leftarrowtail	$\rightarrowtail$	\rightarrowtail	$\downharpoonleft$	\downharpoonleft
$\leftrightharpoons$	\leftrightharpoons	$\rightleftharpoons$	\rightleftharpoons	$\downharpoonright$	\downharpoonright
$\Lsh$	\Lsh	$\Rsh$	\Rsh	$\rightsquigarrow$	\rightsquigarrow
$\looparrowleft$	\looparrowleft	$\looparrowright$	\looparrowright	$\leftrightsquigarrow$	\leftrightsquigarrow
$\curvearrowleft$	\curvearrowleft	$\curvearrowright$	\curvearrowright		
$\circlearrowleft$	\circlearrowleft	$\circlearrowright$	\circlearrowright		

Таблица 19.27. Символы двоичных операций из пакета AMS

$\dotplus$	\dotplus	$\centerdot$	\centerdot	$\intercal$	\intercal
$\ltimes$	\ltimes	$\rtimes$	\rtimes	$\divideontimes$	\divideontimes
$\Cup$	\Cup or \doublecup	$\Cap$	\Cap or \doublecap	$\smallsetminus$	\smallsetminus
$\veebar$	\veebar	$\barwedge$	\barwedge	$\doublebarwedge$	\doublebarwedge
$\boxplus$	\boxplus	$\boxminus$	\boxminus	$\circleddash$	\circleddash
$\boxtimes$	\boxtimes	$\boxdot$	\boxdot	$\circledcirc$	\circledcirc
$\leftthreetimes$	\leftthreetimes	$\rightthreetimes$	\rightthreetimes	$\circledast$	\circledast
$\curlyvee$	\curlyvee	$\curlywedge$	\curlywedge		

Таблица 19.28. Ограничители из пакета AMS

$\ulcorner$	\ulcorner	$\urcorner$	\urcorner	$\llcorner$	\llcorner	$\lrcorner$	\lrcorner

Таблица 19.29. Буквы греческого и еврейского алфавитов из пакета AMS

$\digamma$	\digamma	$\varkappa$	\varkappa	$\beth$	\beth	$\daleth$	\daleth	$\gimel$	\gimel

Таблица 19.30. Символы двоичных отношений из пакета AMS

$\lessdot$	\lessdot	$\gtrdot$	\gtrdot	$\doteqdot$	\doteqdot or \Doteq
$\leqslant$	\leqslant	$\geqslant$	\geqslant	$\risingdotseq$	\risingdotseq
$\eqslantless$	\eqslantless	$\eqslantgtr$	\eqslantgtr	$\fallingdotseq$	\fallingdotseq
$\leqq$	\leqq	$\geqq$	\geqq	$\eqcirc$	\eqcirc
$\lll$	\lll or \llless	$\ggg$	\ggg or \gggtr	$\circeq$	\circeq
$\lesssim$	\lesssim	$\gtrsim$	\gtrsim	$\triangleq$	\triangleq
$\lessapprox$	\lessapprox	$\gtrapprox$	\gtrapprox	$\bumpeq$	\bumpeq
$\lessgtr$	\lessgtr	$\gtrless$	\gtrless	$\Bumpeq$	\Bumpeq
$\lesseqgtr$	\lesseqgtr	$\gtreqless$	\gtreqless	$\thicksim$	\thicksim
$\lesseqqgtr$	\lesseqqgtr	$\gtreqqless$	\gtreqqless	$\thickapprox$	\thickapprox
$\preccurlyeq$	\preccurlyeq	$\succcurlyeq$	\succcurlyeq	$\approxeq$	\approxeq
$\curlyeqprec$	\curlyeqprec	$\curlyeqsucc$	\curlyeqsucc	$\backsim$	\backsim
$\precsim$	\precsim	$\succsim$	\succsim	$\backsimeq$	\backsimeq
$\precapprox$	\precapprox	$\succapprox$	\succapprox	$\vDash$	\vDash
$\subseteqq$	\subseteqq	$\supseteqq$	\supseteqq	$\Vdash$	\Vdash
$\Subset$	\Subset	$\Supset$	\Supset	$\Vvdash$	\Vvdash
$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\backepsilon$	\backepsilon
$\therefore$	\therefore	$\because$	\because	$\varpropto$	\varpropto
$\shortmid$	\shortmid	$\shortparallel$	\shortparallel	$\between$	\between
$\smallsmile$	\smallsmile	$\smallfrown$	\smallfrown	$\pitchfork$	\pitchfork
$\vartriangleleft$	\vartriangleleft	$\vartriangleright$	\vartriangleright	$\blacktriangleleft$	\blacktriangleleft
$\trianglelefteq$	\trianglelefteq	$\trianglerighteq$	\trianglerighteq	$\blacktriangleright$	\blacktriangleright

По умолчанию цифры и стандартные функции печатаются прямым шрифтом, а буквы выводятся курсивом. Описанные выше команды текстовой моды для изменения шрифта в формулах не пригодны, однако существуют соответствующие

команды для математической моды. Отметим, что для подключения некоторых шрифтов требуется использование специальных пакетов. В левой колонке табл. 19.33 дана команда изменения шрифта, в центральной колонке приводится пример шрифта, а в правой — названия необходимых пакетов.

Таблица 19.31. Сопутствующие символы из пакета AMS

$\hbar$	\hbar	$\hslash$	\hslash	$\Bbbk$	\Bbbk
$\square$	\square	$\blacksquare$	\blacksquare	$\circledS$	\circledS
$\vartriangle$	\vartriangle	$\blacktriangle$	\blacktriangle	$\complement$	\complement
$\triangledown$	\triangledown	$\blacktriangledown$	\blacktriangledown	$\Game$	\Game
$\lozenge$	\lozenge	$\blacklozenge$	\blacklozenge	$\bigstar$	\bigstar
$\angle$	\angle	$\measuredangle$	\measuredangle	$\sphericalangle$	\sphericalangle
$\diagup$	\diagup	$\diagdown$	\diagdown	$\backprime$	\backprime
$\nexists$	\nexists	$\Finv$	\Finv	$\varnothing$	\varnothing
$\eth$	\eth	$\mho$	\mho		

Таблица 19.32. Символы стрелок и бинарных отношений с отрицанием пакета AMS

$\nless$	\nless	$\ngtr$	\ngtr	$\varsubsetneqq$	\varsubsetneqq
$\lneq$	\lneq	$\gneq$	\gneq	$\varsupsetneqq$	\varsupsetneqq
$\nleq$	\nleq	$\ngeq$	\ngeq	$\nsubseteqq$	\nsubseteqq
$\nleqslant$	\nleqslant	$\ngeqslant$	\ngeqslant	$\nsupseteqq$	\nsupseteqq
$\lneqq$	\lneqq	$\gneqq$	\gneqq	$\nmid$	\nmid
$\lvertneqq$	\lvertneqq	$\gvertneqq$	\gvertneqq	$\nparallel$	\nparallel
$\nleqq$	\nleqq	$\ngeqq$	\ngeqq	$\nshortmid$	\nshortmid
$\lnsim$	\lnsim	$\gnsim$	\gnsim	$\nshortparallel$	\nshortparallel
$\lnapprox$	\lnapprox	$\gnapprox$	\gnapprox	$\nsim$	\nsim
$\nprec$	\nprec	$\nsucc$	\nsucc	$\ncong$	\ncong
$\npreceq$	\npreceq	$\nsucceq$	\nsucceq	$\nvdash$	\nvdash
$\precneqq$	\precneqq	$\succneqq$	\succneqq	$\nvDash$	\nvDash
$\precnsim$	\precnsim	$\succnsim$	\succnsim	$\nVdash$	\nVdash
$\precnapprox$	\precnapprox	$\succnapprox$	\succnapprox	$\nVDash$	\nVDash
$\subsetneq$	\subsetneq	$\supsetneq$	\supsetneq	$\ntriangleleft$	\ntriangleleft
$\varsubsetneq$	\varsubsetneq	$\varsupsetneq$	\varsupsetneq	$\ntriangleright$	\ntriangleright
$\nsubseteq$	\nsubseteq	$\nsupseteq$	\nsupseteq	$\ntrianglelefteq$	\ntrianglelefteq
$\subsetneqq$	\subsetneqq	$\supsetneqq$	\supsetneqq	$\ntrianglerighteq$	\ntrianglerighteq
$\nleftarrow$	\nleftarrow	$\nrightarrow$	\nrightarrow	$\nleftrightarrow$	\nleftrightarrow
$\nLeftarrow$	\nLeftarrow	$\nRightarrow$	\nRightarrow	$\nLeftrightarrow$	\nLeftrightarrow

Размер символа в формуле зависит от местоположения, то есть буквы и цифры уменьшаются в индексах, дробях и т. п. Существуют четыре основных стиля форматирования математических формул, которые вводятся следующими командами:

```
\displaystyle
\textstyle
\scriptstyle
\scriptscriptstyle
```

Проиллюстрируем использование этих команд:

```
${\displaystyle Text}, {\textstyle Text}, {\scriptstyle Text}, {\scriptscriptstyle Text}$
```

$\displaystyle Text, \textstyle Text, \scriptstyle Text, \scriptscriptstyle Text$

Таблица 19.33. Команды изменения шрифтов в математической моде

Название шрифта	Пример	Необходимый пакет
\mathrm{ABCdef}	<$Fch19_48.tif>	
\mathit{ABCdef}	<$Fch19_49.tif>	
\mathnormal{ABCdef}	<$Fch19_50.tif>	
\mathcal{ABC}	<$Fch19_51.tif>	
\mathcal{ABC}	<$Fch19_52.tif>	Пакет eual с параметром mathcal
\mathfrak{ABCdef}	<$Fch19_53.tif>	Пакет eufrak
\mathbb{ABC}	<$Fch19_54.tif>	Пакет amsfonts или amssymb

И наконец, в математическую формулу можно вставить фрагмент текста при помощи команды \mbox, оформленный по описанным в предыдущих разделах правилам:

```
\begin{displaymath} \mbox{\textit{Для}} \: \forall \: x \: \exists \: \delta \: \mbox{такое,
\textbf{что}} \end{displaymath}
```

Для $\forall\ x\ \exists\ \delta$ такое, **что**

Степени, индексы, разделители, функции

Для набора степеней и индексов в формулах используются соответственно символы «^» и «_». Если индексом или степенью является выражение, то его надо заключить в фигурные скобки. Если у символа есть верхний и нижний индексы, то их можно указывать в любой последовательности. Для того чтобы верхние и нижние индексы располагались на разных расстояниях от символа, можно использовать «пустые» формулы. Например:

```
$a_1$ \qquad \(x^2\) \qquad $e^{-\alpha t}$ \qquad $\beta^3_{i+j}$ \\* \qquad $e^{x^2} \neq
{e^x}^2$ \qquad $R_j{}^i_{k l}$
```

a_1 x^2 $e^{-\alpha t}$ β^3_{i+j}
$e^{x^2} \neq e^{x2}$ $R_j{}^i_{kl}$

Квадратный корень вводится как \sqrt, а корень степени n генерируется командой \sqrt[n]. Высота и ширина знака корня определяются LaTeX автоматически. Чтобы поставить только знак корня, не продолжая его над последующими символами, используется команда \surd.

Часто размеры знака корня у соседних символов оказываются различными. Для того чтобы таких явлений не было, используются пустые (невидимые) символы — страты, имеющие определенную ширину и высоту:

- \mathstrut — невидимый символ, имеющий высоту круглой скобки;
- \vphantom{math} — символ, имеющий высоту формулы math;
- — пробел ширины выражения math.

Приведем демонстрационный пример:

```
$\sqrt{d^2+x}+\sqrt{y}$ \\* $\sqrt{\mathstrut d^2+x}+\sqrt{\mathstrut y}$ \\* $\sqrt[3]{2}
\: \surd (x^2+y^2)$
```

$\sqrt{d^2+x}+\sqrt{y}$
$\sqrt{\mathstrut d^2+x}+\sqrt{\mathstrut y}$
$\sqrt[3]{2} \: \surd (x^2+y^2)$

Для набора математических функций существует ряд команд, написание которых идентично принятым в математике сокращениям. При печати эти имена стандартных математических функций выводятся прямым шрифтом. Имена основных математических функций приведены в табл. 19.34.

Таблица 19.34. Имена математических функций

log	\log	lg	\lg	ln	\ln
arg	\arg	ker	\ker	dim	\dim
hom	\hom	deg	\deg	exp	\exp
sin	\sin	cos	\cos	tan	\tan
arcsin	\arcsin	arccos	\arccos	arctan	\arctan
cot	\cot	sec	\sec	csc	\csc
sinh	\sinh	cosh	\cosh	tanh	\tanh
coth	\coth	lim	\lim	lim sup	\limsup
lim inf	\liminf	inf	\inf	max	\max
det	\det	gcd	\gcd		

Любую из этих операций можно снабдить верхним или нижним индексом:

```
\[  \lim_{x \rightarrow 0} \frac{\sin x}{x} \qquad \max_{x>0} f(x)  \]
```

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} \qquad \max_{x>0} f(x)$$

Дроби можно писать в одну строку при помощи символа наклонной черты «/», а для дробей на нескольких уровнях применяется команда \frac{верх}{низ}. Для набора столбцов из двух элементов, можно пользоваться командами:

```
{… \choose …} или {… \atop …}
```

Первая команда отличается от второй тем, что столбец заключается в скобки. Приведем пример:

```
Дробь в тексте. Вариант 1: $(x+1)/x$. Вариант~2: $\frac{x+1}{x}$. А вот как она выглядит в
отдельной формуле: \begin{displaymath} (x+1)/x, \qquad \frac{x+1}{x} \end{displaymath}
Иллюстрация двустрочных элементов: \[ {x+1 \choose x}, \quad {x+1 \atop x} \]
```

Дробь в тексте. Вариант 1: $(x+1)/x$.
Вариант 2: $\frac{x+1}{x}$. А вот как она выглядит
в отдельной формуле:

$$(x+1)/x, \qquad \frac{x+1}{x}$$

Иллюстрация двустрочных элементов:

$$\binom{x+1}{x}, \quad {x+1 \atop x}$$

Для сложных дробей с несколькими уровнями вложенности возникает проблема с размерами символов, которую можно разрешить, управляя стилями формулы:

```
Так выглядит формула, набранная обычным образом: \[ \frac{1} {x+ \frac{1} {x+\frac{1} {x+
\frac{x} {x+1} } } } \] А это усовершенствованная формула: \[ \frac{1} {x+ \displaystyle
\frac{1} {x+ \displaystyle \frac{1} {x+ \displaystyle \frac{x}{x+1} } } } \]
```

Так выглядит формула, набранная обычным образом:

$$\frac{1}{x+\frac{1}{x+\frac{1}{x+\frac{x}{x+1}}}}$$

А это усовершенствованная формула:

$$\frac{1}{x+\displaystyle\frac{1}{x+\displaystyle\frac{1}{x+\displaystyle\frac{x}{x+1}}}}$$

Существует ряд команд, генерирующих символы различного размера в зависимости от того, в какой формуле такой символ появляется (в тексте или выделенной формуле). В табл. 19.35 дан полный список символов переменного размера.

Как правило, у таких команд могут быть так называемые пределы (например, индексы суммирования или пределы интегрирования). В исходном файле пределы обозначаются как индексы. Чтобы пределы стояли справа от символа, необходимо использовать команду `\nolimits`, а для того, чтобы пределы всегда стояли над и под символом, — команду `\limits`. Например:

```
Сумма в строчной формуле: $\sum_{i=0}^{\infty} a_i$, а теперь в выделенной в строку:
\begin{equation} \sum_{i=0}^{\infty} a_i \neq \sum\nolimits_{i=-\infty}^{0} a_i \end{equation}
Это пример различного вида пределов интегрирования: \begin{displaymath} \int_0^{\frac{\pi}{2}}
\sin x \: dx = \int\limits_0^{\frac{\pi}{2}} \cos x \: dx = 1 \end{displaymath}
```

Сумма в строчной формуле: $\sum_{i=0}^{\infty} a_i$, а теперь в выделенной в строку:

$$\sum_{i=0}^{\infty} a_i \neq \sum\nolimits_{i=-\infty}^{0} a_i \qquad (1)$$

Это пример различного вида пределов интегрирования:

$$\int_0^{\frac{\pi}{2}} \sin x \, dx = \int\limits_0^{\frac{\pi}{2}} \cos x \, dx = 1$$

Теперь рассмотрим применение разделителей, под которыми понимаются различные скобки. Перечень всех символов, которые LaTeX воспринимает как разделители, дан в таблицах 19.36 и 19.37.

Таблица 19.35. Символы переменного размера

$\sum$	\sum	$\bigcup$	\bigcup	$\bigvee$	\bigvee	$\bigoplus$	\bigoplus
$\prod$	\prod	$\bigcap$	\bigcap	$\bigwedge$	\bigwedge	$\bigotimes$	\bigotimes
$\coprod$	\coprod	$\bigsqcup$	\bigsqcup			$\bigodot$	\bigodot
$\int$	\int	$\oint$	\oint			$\biguplus$	\biguplus

Таблица 19.36. Разделители

(	(	)	)	$\uparrow$	\uparrow	$\Uparrow$	\Uparrow
[	[or \lbrack	]	] or \rbrack	$\downarrow$	\downarrow	$\Downarrow$	\Downarrow
{	\{ or \lbrace	}	\} or \rbrace	$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\langle$	\langle	$\rangle$	\rangle	|	| or \vert	$\|$	\\| or \Vert
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil	$\rceil$	\rceil
/	/	\	\backslash		. (dual. empty)		

Таблица 19.37. Большие разделители

$\lgroup$	\lgroup	$\rgroup$	\rgroup	$\lmoustache$	\lmoustache	$\rmoustache$	\rmoustache
$\arrowvert$	\arrowvert	$\Arrowvert$	\Arrowvert	$\bracevert$	\bracevert		

По умолчанию размер разделителей считается фиксированным, однако часто, особенно в многоуровневых формулах, их размер необходимо изменить. Для автоматического изменения размера разделителей до высоты заключенного между ними выражения используются команды \left и \right. Эти команды должны стоять непосредственно перед разделителем. В некоторых случаях возникает необходимость ручного задания размеров скобок. Для этого существует набор команд, которые надо ставить перед разделителем:

```
\bigl        \Bigl    \biggl     \Biggl
\bigr        \Bigr  \biggr  \Biggr
```

Команды в первой строке предназначены для левых разделителей, а во второй — для правых. Команды перечислены в порядке возрастания размера ограничителя. Приведем пример, иллюстрирующий работу с размерами скобок:

```
Вариант 1: $(\frac{(a+1)^2}{b_3})^4$ \\  Вариант 2: $\left (\frac{(a+1)^2}{b_3} \right)^4$
\\  Вариант 3: $\biggl (\frac{(a+1)^2}{b_3} \biggr)^4$
```

Вариант 1: $(\frac{(a+1)^2}{b_3})^4$

Вариант 2: $\left(\frac{(a+1)^2}{b_3}\right)^4$

Вариант 3: $\biggl(\frac{(a+1)^2}{b_3}\biggr)^4$

Команды \overbrace и \underbrace генерируют длинные фигурные скобки соответственно над и под математическим выражением:

```
$\underbrace{a_1+a_2 h + a_3 h^2 + \dots}_{n\: \mbox{\scriptsize штук}} $
```

$\underbrace{a_1+a_2 h + a_3 h^2 + \dots}_{n\: \mbox{\scriptsize штук}}$

Матрицы и системы уравнений

Ранее обсуждался вопрос о наборе формулы в случае, когда она занимает только одну строку или находится в тексте. Однако часто формула состоит из нескольких строк (например, матрица или система уравнений). На этот случай в пакете существуют специальные команды.

Для создания матрицы используется окружение array:

```
\begin{array}[pos]{cols}
...
\end{array}
```

Здесь обязательным является параметр cols, который указывает число столбцов в матрице и способ их позиционирования. Каждому столбцу соответствует одна буква: l — выравнивание по левой границе, r — выравнивание по правой границе, c — выравнивание по центру. Вертикальное позиционирование для всех строк матрицы можно указать при помощи необязательного параметра, который может принимать следующие значения: t — выравнивание по верхней строке, c — выравнивание по центру строки, b — выравнивание по нижней строке.

Строки матрицы разделяются командой перехода на новую строку \\, а столбцы разделяются символами «&». Если матрица окружена разделителями (например, скобками), то для автоматического выбора их размера стоит воспользоваться командами \left и \right. Заметим, что перенос строки не ставится после последней строки и знак «&» не должен стоять последним в строке. Приведем пример, иллюстрирующий работу с матрицами:

```
\begin{equation} \begin{array}[t]{lcr} -y& -x& 0 \\ x+1& -z & x-y \\ -y-z& 0
& 2z  \end{array} \end{equation}
```

$$\begin{array}[t]{lcr} -y & -x & 0 \\ x+1 & -z & x-y \\ -y-z & 0 & 2z \end{array} \qquad (1)$$

```
\begin{displaymath} \left(  \begin{array}{ccc}  1 & 2 & 3 \\  3 & 4 & 5 \\  6 & 7 & 8
\end{array}  \right)  \end{displaymath}
```

$$\left(\begin{array}{ccc} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{array} \right)$$

С помощью окружения array довольно просто набирать системы уравнений, что видно из следующего примера:

```
\[ \left\{ \begin{array}[c]{rcl} x^2-y^2 & = & 0 \\  x+y & = & 12 \end{array} \right. \]
```

$$\left\{ \begin{array}{rcl} x^2-y^2 & = & 0 \\ x+y & = & 12 \end{array} \right.$$

Отметим, что в этом примере мы отвели по столбцу на левую часть системы уравнений, на знак равенства и правую часть уравнений, причем выравнивание символов выбрано соответственно по правому краю, по центру и по левому краю столбца.

Еще одним способом набора многострочных формул является использование следующего окружения:

```
\begin{eqnarray} … \end{eqnarray}
```

Это окружение создает выделенные формулы из нескольких строк, которые выровнены в три колонки, подобно тому как это было сделано в предыдущем примере. Внутри окружения должна находиться последовательность строк, причем все строки, за исключением последней, должны заканчиваться командой `\\`. Каждая строка состоит из трех колонок, разделенных символом «&». По умолчанию все колонки нумеруются как формулы. Для того чтобы строка не нумеровалась, следует использовать команду `\nonumber`. На любую строку можно ссылаться как на отдельную формулу, пометив ее командой `\label`. При форматировании очень длинных формул полезна команда `\lefteqn{math}`. Приведем примеры задания многострочных формул:

```
\begin{eqnarray}          \sin x & = & x - \frac{x^3}{3!} + \nonumber \\ & & +
\frac{x^5}{5!} -      \frac{x^7}{7!} + \frac{x^9}{9!} +O(x^{12})    \end{eqnarray}
```

$$\begin{aligned} \sin x &= x - \frac{x^3}{3!} + \\ &\quad + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + O(x^{10}) \qquad (1) \end{aligned}$$

```
\begin{eqnarray}          f(x) & = & \cos x \\       f'(x) & = & - \sin x \\    \int f(x)\: dx
& = & \sin x \nonumber    \end{eqnarray}
```

$$\begin{aligned} f(x) &= \cos x \qquad (2) \\ f'(x) &= -\sin x \qquad (3) \\ \int f(x)\, dx &= \sin x \end{aligned}$$

Обратим внимание на удобство использования знака «&» для набора многострочной формулы. Используя его в первом примере, мы расположили текст формулы на второй строке правее знака равенства первой строки.

Графика, таблицы, оглавление, библиография

Оптимальным образом размещать рисунки и таблицы в документе помогает введение плавающих объектов, которые должны полностью находиться на одной странице. Плавающий объект автоматически размещается пакетом в том месте, где он помещается целиком. Существуют два стандартных окружения, позволяющие формировать такие объекты:

```
\begin{figure}[loc] … \end{figure}
\begin{table}[loc] … \end{table}
```

Здесь необязательный параметр `loc` определяет способ размещения плавающего объекта и может состоять из последовательности следующих символов: `h` означает, что объект надо разместить после текущей строки, `p` — разместить на отдельной странице, `t` — наверху страницы, `b` — внизу страницы. По умолчанию пакетом устанавливается последовательность `tbp`. Кроме того, существует параметр `(!)`, который ослабляет ограничения, если стоит вместе с одним из вышеописанных параметров.

Для нанесения подписей к плавающим объектам используется команда

```
\caption[entry]{head}
```

Здесь необязательный параметр entry используется для составления списка рисунков или таблиц, а параметр head определяет текст подписи. Этот текст не должен быть разделен на абзацы и не может превышать двухсот символов.

Иногда плавающие объекты автоматически располагаются на нежелательной странице. В этом случае такое их размещение можно запретить при помощи команды

```
\suppressfloats[noloc]
```

Параметр noloc может принимать значения t или b.

Рисование средствами LaTeX

В LaTeX включен набор команд, которые позволяют создавать несложные рисунки или чертежи. Для этого существует следующее окружение:

```
\begin{picture}(width,height)(x0,y0)
 ...
\end{picture}
```

Первая пара параметров этого окружения обязательна и определяет соответственно высоту и ширину рисунка. Единица измерения LaTeX задается переменной unitlength, которая по умолчанию имеет значение 1pt. Вторая (необязательная) пара параметров задает координаты левого нижнего угла рисунка. При отсутствии этих параметров считается, что левый нижний угол имеет координаты (0,0).

Внутри окружения можно использовать специальные графические команды, некоторые из которых перечислены в табл. 19.38.

Таблица 19.38. Графические команды пакета LaTeX

Команда	Описание
\put(x,y){object}	Определяет точку привязки с координатами x и y графического объекта object на рисунке. Объектом может быть текст, как в блоках, так и без них, или графические команды
\line(a,b){length}	Проводит отрезок заданной длины length из текущей точки рисунка. Первый параметр, состоящий из пары целых чисел из интервала от –4 до 4, задает наклон линии. Отношение этих чисел определяет тангенс угла наклона отрезка
\linethickness{val}	Определение толщины вертикальных и горизонтальных линий в любых единицах измерения
\thinlines	Назначает ширину всех линий тонкой
\thicklines	Назначает ширину всех линий толстой
\qbezier[num](xa,ya)(xb,yb)(xc,yc)	Рисование кривой Безье, проходящей через три точки. Необязательный параметр num задает число выводимых точек кривой, в случае его отсутствия выводится непрерывная кривая. Координаты точек задаются в круглых скобках
\vector(xs,ys){dx}	Рисование стрелок. Назначение параметров аналогично команде line

продолжение ⇨

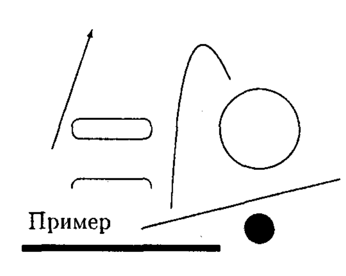
Таблица 19.38 (*продолжение*)

Команда	Описание
\circle{diam}	Рисование окружности радиуса diam с центром в текущей точке
\circle*{diam}	Рисование круга черного цвета, радиуса diam с центром в текущей точке
\oval(width,height) [part]	Рисование овала. В качестве параметра part указывается, какую часть овала рисовать: t — верхнюю половину, b — нижнюю половину, r — правую половину и l — левую половину; width — горизонтальный размер, а height — вертикальный

Пример использования некоторых описанных команд:

```
\begin{picture}(200,200) Пример
\put(-30,40){\vector(1,3){20}} \qbezier(30,10)(35,130)(60,75)
\put(15,0){\line(4,1){100}} \put(0,20){\oval(40,10)[t]}
\put(0,50){\oval(40,10)} \put(75,50){\circle{75}}
\linethickness{3pt} \put(-45,-10){\line(1,0){100}}
\put(75,0){\circle*{75}} \end{picture}
```

В результате получим следующий рисунок:



Графика для LaTeX создавалась давно и, естественно, уже устарела, однако существует возможность включения в документ рисунков, созданных другими, более развитыми средствами.

Включение графических файлов

LaTeX позволяет вставлять рисунки, находящиеся в графических файлах наиболее распространенных форматов (о форматах графических файлов см. раздел «Графические системы и файлы» этой главы), причем набор допустимых форматов зависит от операционной системы. Поэтому желательно использовать аппаратно-независимые форматы, такие как PostScript. Существует целый ряд расширений LaTeX для включения графических файлов, но мы остановимся на распространенном пакете графики graphicx. Это более мощный вариант пакета graphics. Вставка рисунка из графического файла в пакете graphicx осуществляется командой

```
\includegraphics[optionlist]{filename}
```

Обязательный параметр filename определяет имя графического файла с расширением. Аргумент optionlist представляет собой список параметров, которые управ-

ляют характеристиками изображения. Эти параметры представляют собой выражения типа равенств, в левой части которых стоит параметр, а в правой – его значение. В табл. 19.39 перечислены основные допустимые параметры.

Таблица 19.39. Параметры команды импортирования графических файлов

Параметр	Описание
bb	Определяет размеры рисунка: четыре числа, которые разделены пробелами. Первые два числа определяют координаты нижнего левого угла, а два других задают координаты правого верхнего угла
bbllx, bblly	
bburx, bbury	Эти две пары параметров задают соответственно координаты левого нижнего и правого верхнего углов рисунка
natwidth, natheight	Определяют ширину и высоту рисунка в заданных единицах измерения
clip	Если значение этого параметра равно true, то выходящая за границы видимой области часть рисунка отсекается. Когда значение параметра равно false, отсечения не происходит
viewport	Видимая часть рисунка, которая задается двумя парами координат: нижнего левого и верхнего правого углов видимой области
trim	Определяет видимую область рисунка. Например, если trim=1mm 2mm 3mm 4mm, то от рисунка будет отрезано 1 мм слева, 2 мм снизу, 3 мм справа и 4 мм сверху
draft	Устанавливает черновой режим импортирования, когда вместо рисунка выводится рамка по размерам рисунка, а внутри рамки дается имя файла вместо самого рисунка
scale	Изменение размера рисунка, например если указано scale=3, то рисунок будет увеличен в 3 раза
width	Ширина рисунка при печати
height	Высота рисунка при печати
origin	Определяет положение центра вращения рисунка. Вертикальная и горизонтальная координаты центра задаются дискретно и могут принимать следующие значения: горизонтальная координата — l (слева), c (в центре), r (справа); вертикальная координата — t (сверху), c (в центре), b (внизу). Например, origin=rb задает вращение вокруг правого нижнего угла
angle	Поворот изображения на заданный параметром angle угол вокруг точки, заданной параметром origin (по умолчанию вокруг точки привязки). Углы задаются в градусах
keepaspectratio	Если этот параметр имеет значение true, то при изменении размеров рисунка сохраняются пропорции

Заметим, что обычно рисунки помещают внутри окружения

```
\begin{figure} … \end{figure}
```

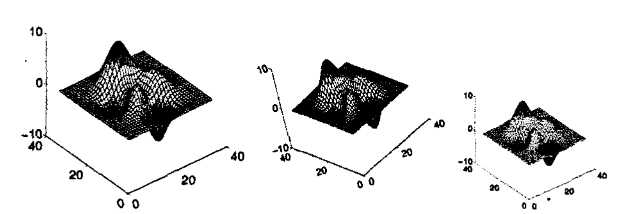
что позволяет легко делать подписи к рисункам и автоматически их нумеровать.

Приведем пример с перечисленными командами, используя в качестве графических файлов рисунки, подготовленные средствами пакетов MATLAB и Maple.

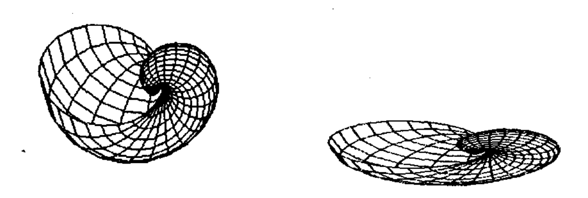
В первом случае импортируется векторная графика PostScript-файла (файл test.eps), из которого LaTeX сам берет информацию о размере изображения. Отметим, что при вводе растровой графики, например, bmp-файлов, необходимо указывать размеры рисунка (см. последние две команды примера):

```
\documentclass[12pt]{article}
\usepackage{graphicx}
\begin{document}
% импортирование векторной графики
\includegraphics[width=3cm]{test.eps}
\includegraphics[angle=15,height=3cm]{test.eps}
\includegraphics[height=2cm, width=2cm]{test.eps}
\newpage
% импортирование растровой графики
\includegraphics[bb = 0 0 3.5cm 3cm]{test.bmp}
\includegraphics[bb = 0 0 130pt 40pt]{test.bmp}
\end{document}
```

В результате на первой странице документа будет помещен первый рисунок:



А на следующей странице — второй:



Верстка таблиц

Окружение tabular применяется для создания таблиц, содержащих текст, причем как с рамками, так и без них. Окружение имеет вид:

```
\begin{tabular}{spec} … \end{tabular}
```

В качестве спецификаций spec применяются следующие обозначения: l — для колонки с выравниванием влево, r — для колонки с выравниванием вправо, c — для центрированного текста, p{width} — для колонки, содержащей текст с разрывами линий, символ «|» используется для проведения вертикальных линий, а команда hline — для горизонтальных. Переход к новой строке таблицы осуществляется командой \\, а для разделения колонок применяется амперсант «&». Пробелы в начале и конце графы таблицы игнорируются. Для вставки текста text между столбцами существует команда @{text}.

Для создания надписей, которые охватывают несколько колонок, используется команда \multicolumn. У этой команды три обязательных параметра: количество охватываемых колонок, тип выравнивания текста в этой графе и сам текст.

Иногда в таблице надо провести короткую горизонтальную линию. Это можно сделать командой \cline, имеющей обязательный параметр — номера первой и последней подчеркиваемых колонок. Команда \vline проводит вертикальную линию на всю высоту строки.

Приведем примеры различных таблиц:

```
\begin{tabular}{|l|c|r|}
\hline
Глава & Пакет & Назначение \\ \hline
1 & Maple & Символьные вычисления\\
2 & Matlab & Численный анализ\\
3 & \LaTeX & Верстка текстов\\ \hline
\end{tabular}
```

Результат:

Глава	Пакет	Назначение
1	Maple	Символьные вычисления
2	Matlab	Численный анализ
3	LaTeX	Верстка текстов

Пример более сложной таблицы:

```
\begin{tabular}{| l | r@{ m} | r@{ cm}|}
\hline \multicolumn{3}{|c|}{Английские меры длины} \\
\hline & \multicolumn{2}{|c|}{Соответствие} \\ \cline{2-3}
Единицы & \hspace{1.5cm} & \\ \hline
Фут & 0.3 & 30.5 \\ Ярд & 0.9 & 91.44 \\
Миля & 1609 & 160900\\
\hline
\end{tabular}
```

Результат:

Английские меры длины		
	Соответствие	
Единицы	m	cm
Фут	0.3 m	30.5 cm
Ярд	0.9 m	91.44 cm
Миля	1609 m	160900 cm

Для автоматической нумерации и удобства создания подписей таблицы обычно размещаются внутри окружения

```
\begin{table} … \end{table}
```

Оглавления

Под оглавлениями здесь понимаются оглавление документа, а также перечни рисунков и таблиц, которые LaTeX позволяет автоматически генерировать. В процессе работы пакет автоматически собирает информацию о разбивке документа на

разделы (секции) и размещении плавающих объектов. Эта информация записывается в служебные файлы в момент обработки команды \end{document}. Поэтому для того, чтобы оглавления были сгенерированы правильно, следует входной файл обработать транслятором по крайней мере три раза. Считывают из файлов и печатают оглавление, списки таблиц и рисунков соответственно команды:

```
\tableofcontents
\listoffigures
\listoftables
```

Информацию об оглавлении формируют команды секционирования (разбивка документа на разделы, главы и пр.). Информация о рисунках и таблицах задается командой \caption. В случае нестандартных ситуаций можно записать в оглавлении дополнительные пункты при помощи команды

```
\addcontentsline{ext}{unit}{entry}
```

Здесь параметр ext определяет, в какое оглавление надо добавить пункт: toc — оглавление документа, lof — список рисунков, lot — список таблиц. В зависимости от значения ext параметр unit может принимать следующие значения: toc — part, chapter и т. д., lof — figure, lot — table. Последний параметр entry задает сам текст строки в оглавлении и, в свою очередь, может содержать некоторые команды.

Существует еще одна команда, которая добавляет текст в оглавление:

```
\addtocontents{ext}{text}
```

Первый параметр аналогичен рассмотренному для предыдущей команды, а вторым параметром является добавляемый текст.

Команда contentsname определяет заголовок оглавления.

Библиография и алфавитный указатель

Библиография — важный компонент любой статьи или книги. LaTeX позволяет легко готовить списки литературы, состоящие из разнообразных источников и автоматически расставлять ссылки на них в тексте документа. Для печати списка цитируемой литературы в LaTeX существует окружение

```
\begin{thebibliography}{par}
\end{thebibliography}
```

Параметр par определяет сдвиг левой границы печати списка литературы на ширину текста, стоящего в качестве этого параметра. Номера источников ставятся на месте образовавшегося пробела, потому задаваемая ширина отступа не должна быть меньше, чем ширина печати максимального номера. Внутри окружения источники перечисляются при помощи команды

```
\bibitem[lab]{key}
```

После этого следует непосредственно текст библиографического источника. Соответствующая запись будет помечена символами lab, а ссылаться на источник можно будет по метке key. При отсутствии метки lab вместо нее будет печататься порядковый номер источника, который хранится в счетчике enumiv. Для ссылок на источники в тексте документа применяется команда

```
\cite[text]{key}
```

Результатом этой команды будет печать символов lab или номера источника с меткой key, заключенных в квадратные скобки. Если одной командой нужно сослаться на разные источники, то соответствующие метки перечисляются через запятую. Необязательный параметр text служит для уточнения ссылки текстовым комментарием. Приведем пример документа, содержащего список литературы:

```
\documentclass[12pt,russian]{article}
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\textwidth=12cm
\begin{document}
Первой книгой по {\TeX} была книга \cite{Knut}, а по {\LaTeX} – книга \cite{Lamport}. Сейчас существует много литературы как о {\TeX}, так и о {\LaTeX} (см. \cite[и др.]{Knut,Lamport}).
\begin{thebibliography}{99999}
\bibitem{Knut} D. E. Knuth. The TeX book Volume A of Computers and Typesetting, Addison-Wesley Publishing Company 1984.
\bibitem[\LaTeX]{Lamport} L. Lamport. LaTeX: A Document Preparation System, Addison-Wesley, Reading, Massachusetts, second edition, 1994.
\end{thebibliography}
\end{document}
```

Результат:

Первой книгой по TeX была книга [1], а по LaTeX – книга [LaTeX]. Сейчас существует много литературы как о TeX, так и о LaTeX (см. [1, LaTeX, и др.]).

Список литературы

[1] D. E. Knuth. The TeX book Volume A of Computers and Typesetting, Addison-Wesley Publishing Company 1984.

[LaTeX] L. Lamport. LaTeX: A Document Preparation System. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

В настоящее время распространено создание баз данных источников по различным областям наук. Для этого используется LaTeX и программа BibTeX, которая позволяет накапливать библиографию в отдельных bib-файлах. Для занесения и извлечения информации из файлов BibTeX имеет свой язык. Это очень удобно для работы, но здесь не будем останавливаться на описании BibTeX, укажем только, что работа с ней описана в книге [32].

Составлением алфавитных указателей занимается специальная программа makeindex, которая входит в набор LaTeX. Сначала нужно подключить пакет makeidx командой

```
\usepackage{makeidx}
```

и в преамбуле документа выполнить команду

```
\makeindex
```

Элементы алфавитного указателя задаются при помощи команды

```
\index{key}
```

Здесь key определяет элемент алфавитного указателя. Например, чтобы внести в алфавитный указатель слово «LaTeX», необходимо в нужном месте исходного текста (там, где находится слово LaTeX) поставить команду index{ LaTeX}.

После обработки исходного файла транслятором будет создан файл с именем исходного файла и расширением .idx, в который записываются последовательно все элементы алфавитного указателя с номерами соответствующих страниц. Чтобы расположить ссылки в алфавитном порядке, этот файл надо обработать программой makeindex, то есть запустить эту программу, указав в качестве параметра имя исходного файла. Результатом будет файл с расширением .ind. Чтобы вставить подготовленный алфавитный указатель в свой документ, нужно поместить в тексте команду

```
\printindex
```

Отметим, что для составления качественного алфавитного указателя этого недостаточно, так как программа размещает компоненты указателя не в алфавитном порядке, а по мере встречи ссылок в исходном документе.

Программирование в LaTeX

В пакете LaTeX имеется довольно мощный набор команд, которые позволяют значительно облегчить верстку как простых, так и сложных документов. Можно переопределять уже существующие команды и окружения, а можно писать новые. Простым и эффективным примером служит определение новых команд для сокращения стандартных имен.

Создание собственных команд, окружений и структур

Для определения новых команд в LaTeX имеется команда

```
\newcommand{name}[num]{definition}
```

Здесь обязательными являются первый и третий параметры, которые соответственно задают имя и содержание новой команды. Необязательный параметр num определяет количество параметров команды, которое может изменяться от 1 до 9. Параметры в тексте определения команды обозначаются двумя символами: «#» и номером параметра.

Для переопределения существующих команд предназначена команда

```
\renewcommand{name}[num]{definition}
```

Смысл параметров аналогичен описанным для команды \newcommand.

Удобно для часто встречающегося сочетания слов или формулы определить команду, генерирующую такой текст. Приведем примеры:

```
\newcommand{\o}{ортогональн}  Набор {\o}ых полиномов удовлетворяет условию {\o}ости.
$\omega$ - полнота.    \renewcommand{\omega}{Омега} \omega - полнота.
```

Набор ортогональных полиномов удовлетворяет условию ортогональности.
ω - полнота. Омега - полнота.

```
\newcommand{\F}[2]{ \frac{\sqrt{#2}}{\sqrt {({#1})^2+({#2})^2}} \\ }  $\F{1}{2}$ $\F{a+b}{c+d}$
```

$$\frac{\sqrt{2}}{\sqrt{(1)^2+(2)^2}}$$
$$\frac{\sqrt{c+d}}{\sqrt{(a+b)^2+(c+d)^2}}$$

Для описания нового окружения и изменения уже существующего используются соответственно команды:

```
\newenvironment{name}[num]{before}{after}
\renewenvironment{name}[num]{before}{after}
```

Имя окружения задается обязательным параметром name, а необязательный параметр num определяет число формальных параметров окружения. Параметр before здесь обозначает набор команд, выполнение которых предшествует анализу текста находящегося внутри окружения, а символами after — команды, выполняющиеся по завершению анализа. Отметим, что формальные параметры окружения могут находиться только в группе команд before.

Приведем пример:

```
\newenvironment{cc}[1]{\begin{center} #1 } {\end{center}}

В первый раз текст выведем обычным шрифтом: \begin{cc}{} Пример окружения, созданного
пользователем \end{cc}
А второй раз наклонным шрифтом, используемым в примечаниях: \begin{cc}{\itshape
\footnotesize} Пример окружения, созданного пользователем \end{cc}
При выходе из окружения все установки восстанавливаются.
```

Результат:

В первый раз текст выведем обычным шрифтом:

Пример окружения, созданного пользователем

А второй раз наклонным шрифтом, используемым в примечаниях:

Пример окружения, созданного пользователем

При выходе из окружения все установки восстанавливаются.

При наборе математических статей часто приходится использовать теоремы, леммы и другие формулировки. Для создания подобной структуры в LaTeX есть команда, размещаемая в преамбуле документа:

```
\newtheorem{name}[counter]{text}[section]
```

Обязательные параметры команды name и text определяют соответственно имя объекта и слово, с которого будет начинаться текст. Параметры в квадратных скобках не являются обязательными и используются для нумерации. Параметр counter привязывает номер описываемого объекта к номеру уже существующего, а пара-

метр section добавляет перед номером структуры (теоремы, леммы, предложения) номер соответствующего раздела. Приведем пример документа, включающего определение двух новых определений:

```
\documentclass[12pt,russian]{article}
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\newtheorem{t1}{Предложение}[section]
\newtheorem{t2}[t1]{Гипотеза}
\begin{document}
\section{Пример}
Демонстрация описания определений.
\begin{t1} Использование структур типа "теорема" бывает удобным. \end{t1}
\begin{t2} Их использование иногда бывает трудоемким. \end{t2}
\end{document}
```

Результат:

1 Пример

Демонстрация описания определений.

Предложение 1.1 *Использование структур типа 'теорема' бывает удобным.*

Гипотеза 1.2 *Их использование иногда бывает трудоемким.*

Создание и изменение счетчиков

Автоматическую нумерацию можно легко организовывать при помощи счетчиков. Счетчик — это специальная переменная, принимающая целые значения. Счетчику можно присваивать значение и изменять его, выводить его значение на печать и организовывать с его помощью автоматическую генерацию ссылок. Мы уже касались в разделе этой главы «Ссылки и нумерация» стандартных счетчиков. Здесь речь пойдет о создании новых счетчиков и работе с их значениями.

Каждый счетчик имеет свое имя и создается с помощью команды

```
\newcounter{newcout}{oldcount}
```

Здесь newcout — имя вновь организуемого счетчика, а oldcount — имя уже существующего счетчика. Второй параметр этой команды не является обязательным и означает то, что вновь организуемый счетчик обнуляется при каждом наращивании счетчика oldcount. Другими словами, вновь организуемый счетчик является подчиненным по отношению к счетчику oldcount. Примером подчиненных счетчиков являются номера глав, секций и т. д.

Для изменения значения существующего счетчика используются команды:

```
\setcounter{name}{num}
\addtocounter{name}{num}
```

Первая команда устанавливает счетчику с именем name значение num, а вторая увеличивает значение счетчика name на величину num.

Существуют еще две команды, которые увеличивают значение счетчика на единицу и обнуляют все подчиненные счетчики (например, счетчик подразделов является подчиненным счетчику раздела):

```
\refstepcounter{count}
\stepcounter{count}
```

Вторая команда применяется реже, так как с ее помощью нельзя организовать автоматические ссылки.

Как стандартные, так и созданные пользователем счетчики могут выводиться арабскими, римскими цифрами и символами латинского алфавита. По умолчанию значение счетчика выводится арабскими цифрами. Для того чтобы выводить значение счетчика на печать арабскими и римскими цифрами, а также буквами латинского алфавита, используются соответственно команды:

```
\arabic{count}
\roman{count} и \Roman{count} для малых и больших римских цифр.
\alph{count} и \Alph{count} для малых и больших латинских букв.
```

Приведем пример документа, включающего определение новых счетчиков:

```
\documentclass[12pt,russian]{article}
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\newcounter{c1} \newcounter{c2}[c1]
\begin{document} \noindent
\setcounter{c1}{1}Значение счетчика c1 = \arabic{c1}.
\addtocounter{c2}{3} Значение счетчика c2 = \arabic{c2}.
\refstepcounter{c1}Значение счетчика c1 равно \Roman{c1}, а
значение счетчика c2~=~\arabic{c2}.
\end{document}
```

Результат:

Значение счетчика c1 = 1. Значение счетчика c2 = 3.
Значение счетчика c1 равно II, а значение счетчика
c2 = 0.

Опишем теперь применение счетчиков для организации автоматических ссылок. В следующем документе определим новую команду, которая с нового абзаца будет печатать слово «Задача» жирным шрифтом, изменять номер задачи и выводить его:

```
\documentclass[12pt,russian]{article}
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\newcounter{z}
\newcommand{\zdch}
  {\par\textbf{Задача \addtocounter{z}{1}\arabic{z}. }}
\begin{document} \noindent \zdch Решить \ldots \zdch Вычислить \ldots \zdch Доказать \ldots
\end{document}
```

Результат:

Задача 1. Решить ...
Задача 2. Вычислить ...
Задача 3. Доказать ...

Обработка ошибок

При наборе текстов неизбежно возникают ошибки. При трансляции документа все сообщения об ошибках и предупреждения о неточностях показываются на экране и, кроме того, записываются пакетом в файл протокола (расширение .log). Предупреждения не вызывают прерывания процесса трансляции, а при обнаружении синтаксической ошибки трансляция останавливается и на экран выводится сообщение об ошибке.

При трансляции LaTeX-файла возможны ошибки двух типов: ошибки пакета LaTeX и ошибки языка TeX, то есть ошибки более низкого уровня. Если транслятор обнаруживает ошибку, то печатает ее тип (LaTeX-error или TeX-error), номер строки исходного файла, в которой, по мнению пакета, находится ошибка, и само ее содержание. Сообщение заканчивается знаком вопроса. При этом трансляция приостанавливается и пакет переходит в режим ожидания реакции пользователя.

Ошибку можно проигнорировать, нажав клавишу Enter. В этом случае LaTeX сам попробует ее исправить по своему разумению. Набрав <h> и нажав Enter, можно посмотреть информацию об ошибке на английском языке. Если ошибок много, можно при остановке трансляции ввести с клавиатуры <q> и Enter, чтобы трансляция продолжалась далее без остановок, а затем посмотреть все сообщения об ошибках в файле с расширением .log. Ошибки можно исправлять в диалоговом режиме. Для этого при остановке необходимо ввести <i> и Enter, после чего появится предложение <insert>, куда нужно ввести исправленную команду. Но затем все равно необходимо исправить ошибку в исходном файле. И наконец, можно прекратить трансляцию исходного файла, нажав клавиши <x> и Enter.

Мы не будем приводить список возможных сообщений об ошибках с их переводом на русский язык. Эту информацию можно найти в книгах [30, 32, 33]. Заметим только, что TeX — система программирования, а поиск ошибок — непременный атрибут всякого программирования. Здесь квалификация приходит с опытом, набираться которого лучше начиная с небольших текстов.

20 ГЛАВА Редакторы и стандарты

В этой главе приводится информация справочного характера, которая полезна при использовании в научных публикациях результатов, полученных с помощью Maple или MATLAB, и при наборе текстов в системе верстки LaTeX. Сначала дано короткое описание пакета MikTeX и редактора WinEdt вместе с инструкцией по их установке. Эти программы создают удобную среду для работы с TeX. Затем описано создание и использование PostScript-файлов, которые являются стандартными для многих графических систем и представления публикаций в Интернете. В следующих двух разделах главы обсуждаются набор математических текстов в редакторе MS Word и преобразование документов из различных форматов в другие. Заключительный раздел посвящен форматам графических файлов.

Пакет MikTeX

MikTeX является версией пакета TeX, специально адаптированной для операционной системы Windows. Вместе с редактором WinEdt он формирует удобную среду для верстки научных публикаций в среде LaTeX на персональных компьютерах с операционной системой Windows. В предыдущей главе при описании LaTeX мы использовали именно этот пакет, поскольку он поддерживает все возможности TeX, в том числе: компиляцию файлов TeX, включение в текст графических объектов, выделение текста цветом, использование различных шрифтов и автоматическую их генерацию, все стандартные макропакеты, в том числе LaTeX, просмотр и преобразование полученных dvi-файлов в различные форматы, возможности набора текста практически на всех языках, включая автоматическую расстановку переносов и т. д. Пакет включает следующие компоненты:

- TeX 3.14159 — классический компилятор с языка TeX;
- e-TeX — расширенная и усовершенствованная версия TeX;
- Yap — программа просмотра dvi-файлов;
- pdfTeX — программа генерации файлов в формате PDF из документов TeX;

- METAFONT — программа генерации шрифтов;
- dvips — программа преобразования dvi-файлов в PostScript-файлы;
- TtH — программа перевода документов TeX в HTML-документы;
- MakeIndex — программа для создания алфавитных указателей;
- BibTeX — программа для создания библиографических ссылок;
- AMS-LaTeX, Babel, PSNFSS, ... — стандартные пакеты LaTeX;
- TeXware, METAFONTware, PSutils, ... — различные служебные программы.

Пакет постоянно совершенствуется, и его последнюю версию (на момент написания книги это версия 2.0) можно бесплатно получить через Интернет в CTAN (см. главу 21 раздел «TeX в Интернете») в каталоге systems/win32/miktex/.

Установка пакета MikTeX

В первую очередь надо развернуть полученный zip-архив (для версии 2.0 его имя — 2.0.zip) с инсталляционным набором файлов на жестком диске во временном каталоге. В результате распаковки во временном каталоге будут содержаться все файлы необходимые для установки. Для полной установки пакета необходимо около 70 Мбайт дискового пространства, а кроме того, в процессе дальнейшей работы MiKTeX генерирует новые шрифты, что может потребовать дополнительного места на диске порядка 2–5 Мбайт.

Для установки нужно из временного каталога запустить программу setupwiz.exe и следовать инструкции по инсталляции. Все инсталляционные установки по умолчанию достаточно разумны и лучше их не менять, кроме, быть может, имени диска, на который будет производиться установка. Далее, для определенности, будем считать, что это диск C.

Первым после запуска инсталляционной программы на экране появляется окно с информацией о версии пакета. Для продолжения работы в этом и других окнах следует нажимать кнопку с надписью Next. В следующем окне можно выбрать тип установки пакета — для индивидуальной или коллективной работы. После этого возникнет окно для указания каталога, в котором будет установлен MikTeX, а затем окно с перечнем компонент пакета, из которого можно выбрать необходимые модули. Как уже отмечалось, в процессе дальнейшей работы пакет будет создавать новые шрифты, и для них можно определить специальный каталог, указав его в следующем окне. Затем программа установки предложит создать дерево файлов TeX, чего не следует делать без уверенности в необходимости данного действия. Три заключительных инсталляционных окна последовательно выведут определенные пользователем параметры установки, предложат добавить строку с путем к файлам MikTeX в autoexec.bat (что рекомендуется сделать) и прочитать информацию о пакете.

В результате установки на диске появятся указанные при инсталляции каталоги с необходимыми для работы MikTeX файлами и возникнет соответствующий пункт в меню программ Windows. Для набора текстов на русском языке, а точнее для расстановки переносов в русских текстах, необходимо провести дополнительные действия. В каталоге C:\texmf\tex\generic\config следует найти файл language.dat и

в любом текстовом редакторе раскомментировать (то есть убрать символ процента в начале строки) строчку:

```
%russian ruhyphen.tex % note: edit ruhyphen.tex for your…
```

Результат должен быть следующим:

```
russian ruhyphen.tex % note: edit ruhyphen.tex for your…
```

После этого для внесения изменений следует обновить установки пакета, запустив из Windows ряд программ:

- Пуск ▸ Программы ▸ MiKTex ▸ Maintenance ▸ Create All Format Files
- Пуск ▸ Программы ▸ MiKTex ▸ Maintenance ▸ Reconfigure
- Пуск ▸ Программы ▸ MiKTex ▸ Maintenance ▸ Refresh Filename Database

Такие действия рекомендуется проводить в дальнейшем при изменении установок или после добавления новых стилевых файлов. После обновления конфигурации стилевые файлы будут доступны во всех рабочих текстах. В противном случае, они будут подключаться только тогда, когда находятся в том же каталоге, что и рабочий текст.



ВНИМАНИЕ Хотя пакет MikTeX распространяется бесплатно, но нормально его установить или изменить конфигурацию можно только в случае, если используемая версия не старше одного года. Поэтому после изменения конфигурации возможен отказ в работе пакета (если он установлен около года назад).

Решить эту проблему можно, скачав новую версию через Интернет или изменив перед установкой и конфигурированием старую системную дату, а затем снова изменив ее на текущую. Заметим, что при первых запусках MiKTeX создает шрифты, и это может занимать довольно длительное время. Шрифты накапливаются в базе данных и при последующей работе все происходит достаточно быстро.

Редактор WinEdt

Удобной и развитой оболочкой для MikTeX является универсальный и чрезвычайно гибкий текстовый редактор WinEdt. Это так называемая условно-бесплатная программа, то есть она свободно распространяется в Интернете, но по истечении определенного срока начинает напоминать о том, что ее необходимо зарегистрировать. Для регистрации необходимо заплатить некоторую сумму для получения пользователем персонального пароля. Разработчики программы просят не пользоваться генераторами кодов, которые написаны хакерами и которые без труда можно найти в Интернете, а поддержать материально развитие WinEdt. Найти редактор можно на сайтах CTAN (см. следующую главу) или по адресу http://www.winedt.com.

Возможности редактора очень широки: он поддерживает одновременную работу с файлами практически любых размеров, обладает развитым и дружественным интерфейсом, во многом специально ориентированным на LaTeX, включает синтаксический анализ текстов, средства макропрограммирования и др. Об интерфейсе редактора можно судить по рис. 20.1, на котором приведено окно WinEdt. Отметим,

что практически все характеристики как интерфейса, так и всех остальных компонентов могут изменяться пользователем, но установки по умолчанию очень разумны. Детальному описанию редактора можно посвятить отдельную книгу, но мы ограничимся короткой инструкцией по установке и некоторыми практическими советами.

Начнем с технических требований к компьютеру. Для работы с WinEdt в среде Windows нужен компьютер следующей конфигурации: процессор Pentium 133МГц (или лучше), 16 Мбайт оперативной памяти (рекомендуется 32 Мбайт), 10 Мбайт дискового пространства. Интерфейс WinEdt требует установки разрешения экрана не менее чем 800×600. При более низком разрешении редактор будет работать со старым интерфейсом — значительно более бедным.

Для установки редактора следует развернуть полученный в Интернете самораспаковывающийся архив (обычно winedt2k.exe) с инсталляционными файлами во временном каталоге и запустить программу setup.exe. Установка не требует особой квалификации, и можно утвердительно ответить на все предлагаемые программой вопросы. После установки появятся указанные при инсталляции каталоги и возникнет соответствующий пункт в меню программ Windows.

Если WinEdt устанавливался после MikTeX (рекомендуется), то MiKTeX для большинства компьютеров подключается автоматически. Если этого не произошло, то следует воспользоваться меню редактора WinEdt Options ▸ Configuration ▸ Tex. Если предполагается использовать программу gsview для просмотра PostScript-файлов и программу Adobe Acrobat Reader для просмотра PDF-файлов, то лучше всего установить эти программы перед инсталляцией WinEdt. В противном случае не произойдет их автоматического подключения к WinEdt и придется самостоятельно позаботиться об их подключении из пункта меню: Options ▸ Menu Setup ▸ &Accessories.

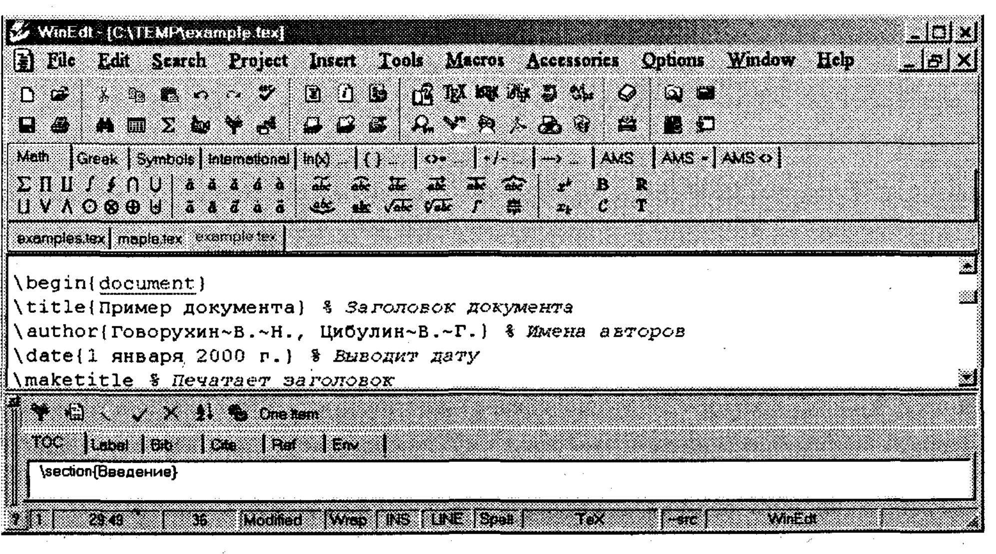


Рис. 20.1. Окно редактора WinEdt

При работе с русскими текстами могут возникнуть некоторые проблемы, в частности после инсталляции редактора обычно установлена латинская кодировка шрифта для отображения текстов на экране. Это легко исправить, обратившись из меню WinEdt к пункту Options ▸ Preferences ▸ Font ▸ Font, а затем выбрав набор символов кириллицы.

После нормального выполнения всех указанных действий по установке пакета MikTeX и редактора WinEdt можно начинать работать. Схематично работу с LaTeX из WinEdt можно представить следующим образом (см. рис. 20.1): пользователь набирает текст, следуя правилам LaTeX, причем редактор предоставляет многочисленные сервисные возможности и подсказки, а для запуска транслятора или программы просмотра dvi-файлов просто обращается к нужному пункту меню Accessories или соответствующему значку. Также просто можно вызывать программы преобразования dvi-файлов в файлы PostScript и PDF или переводчик LaTeX-текстов в HTML-документы. Как уже отмечалось, практически все пункты меню можно изменять и развивать, например, для того, чтобы транслятор запускался клавишей F9, надо проделать следующие операции: Options ▸ Menu Setup ▸ MainMenu ▸ &Accessories ▸ &Latex ▸ поле ShortCut ▸ F9.

Далее коротко остановимся на некоторых элементах работы с данными программами, причем будем считать, что WinEdt установлен в каталоге C:\Programm Files\WinEdt.

Для синтаксической проверки русских текстов необходимо установить словари русского языка. Необходимые файлы (расширение dic) можно найти в Интернете или на многочисленных компакт-дисках. Для установки файлы словарей нужно скопировать в каталог C:\Programm Files\WinEdt\Dict, затем зайти в меню WinEdt Options, выбрать пункт Dictionary. Установить указатель мыши на последнюю строчку в списке словарей и нажать правую клавишу мыши. Выбрать Insert. В строке Dictionaries набрать имя словаря, например ru, затем в строке Definition набрать `%B\Dict\ru.dic` или зайти в режим просмотра файлов для поиска словаря. Отметить в нужном поле Enabled (активировать словарь) и Modified (если хотите, чтобы он пополнялся), Load on Start, Load on Exit, Add New Words (если хотите добавлять в него новые слова). Выйти и снова зайти в программу WinEdt (словари будут работать только после перезапуска WinEdt).

Программа просмотра dvi-файлов Yap-viewer, которая входит в состав MikTeX, и WinEdt предоставляет удобную возможность работы с LaTeX-текстами. При просмотре dvi-файла можно, нажав на каком-либо месте текста в Yap (то есть в dvi-файле), автоматически попасть в соответствующее место исходного текста (tex-файл) в WinEdt. Эта возможность доступна автоматически, нужно только в системной панели WinEdt (полоска подсказки внизу) установить параметр `-src` (между пунктами TeX и WinEdt). Далее в программе Yap в меню View ▸ Options ▸ Inverse Search в строке Program выбрать WinEdt и убедиться, что Comand Line имеет вид:

```
"c:\Program Files\WinEdt\WinEdt.exe" "[Open("%f");SelLine(%l,8)]"
```

Если это не работает, то можно обратиться к файлам справки обеих программ.

В WinEdt можно компилировать выделенный фрагмент LaTeX-текста, для чего следует отметить нужный текст и обратиться к Accessories ▸ Compile Selected. Мы перечислили только небольшое подмножество возможностей среды работы с LaTeX-текстами на основе пакета MikTeX и редактора WinEdt. За более подробной информацией следует обращаться к документации этих программных продуктов.

Создание и использование PostScript-файлов

Язык PostScript (PS) широко применяется в издательской деятельности, в научном программировании для визуализации результатов и является одним из стандартов хранения математических публикаций в Интернете. Стандарт PostScript был разработан и развивается до сих пор компанией Adobe Corporation (http://www.adobe.com). Первая публикация об этом языке и первый PostScript-принтер появились в 1985 году. В последующие годы язык развивался, так спустя пять лет после появления первой версии появилась спецификация языка PostScript Level 2. На данный момент существует программное обеспечение, позволяющее легко просматривать, печатать и создавать PS-файлы, не имея специального принтера. Коротко опишем сам язык PostScript и его возможности, а затем дадим ряд практических советов по использованию файлов в этом стандарте.

Введение в язык PostScript

PostScript — это язык описания страниц и двумерных изображений, предназначенных для печати на принтерах. Средствами этого языка описываются форматирование страниц, используемые шрифты, графика. Все это делается не зависящими от конкретного устройства, его разрешения и цветовых свойств методами. Объекты имеют координаты в аппаратно-независимой системе координат, которые отображаются в системе координат устройства. Основными объектами являются текст, точка, линия, окружность, кривая. Линии могут иметь разную толщину и быть окрашены в различные цвета. Все объекты могут трансформироваться (переноситься в новое место, масштабироваться, накладываться друг на друга). К рисункам добавляются тексты, написанные разнообразными шрифтами.

PostScript-файл представляет собой программу на собственном алгоритмическом языке. Программа является последовательностью операторов, заданных в постфиксной нотации или обратной польской записи: операнды стоят перед оператором. Интерпретатор языка PostScript, встроенный в принтер (конечно, если в принтере он есть), записывает в стек операнды, а затем выполняет оператор. Команды языка PostScript имеют вид высокоуровневых команд типа «проведи линию из точки с координатами (x1,y1) в точку (x2,y2)». Язык позволяет выполнять вычисления, имеет переменные и обычные для языков программирования возможности: процедуры, средства преобразования текстов, может считывать и записывать файлы и многое другое. PostScript-программы — это обычные текстовые файлы, которые можно обрабатывать стандартными текстовыми редакторами.

PostScript не специфицирует никакой структуры программы: любая последовательность операторов, удовлетворяющая синтаксису и семантике языка, является допустимой PostScript-программой. Для большой программы, описывающей сложный многостраничный документ, часто необходимо выделить ее структуру. Для структурирования программы используются специального вида комментарии: строки начинающиеся с двух знаков % с первой позиции. За этими символами идет ключевое слово. Структура и набор этих специальных комментирующих строк

описаны в стандарте DSC (Document Structure Convention). Использование DSC позволяет структурно обрабатывать PostScript документы, не опускаясь до уровня PostScripta.

Часто для импорта и экспорта PostScript-изображений между приложениями используется формат EPS (Encapsulated PostScript). Инкапсулированный PostScript-файл — это программа на языке PostScript, описывающая изображение одной страницы и использующая DSC соглашения с некоторыми ограничениями. EPS файл может содержать любую комбинацию текста, графики, растровых образов и является окончательной формой презентации, то есть этот файл не может редактироваться при импорте в документ. Однако над ним можно выполнить некоторые глобальные манипуляции, не меняющие его содержимого: сдвиг, поворот, масштабирование. Имеется ряд PostScript-операторов, запрещенных к использованию в EPS файлах, и ряд требований к формату файла. Такими являются операторы, очищающие стеки, страницу, инициализирующие графическое состояние. Сам файл должен быть 7-битным ASCII, строки не должны превышать 255 символов и т. д. EPS-программа после себя должна оставить стеки операторов и все параметры в таком же состоянии, как до своего выполнения.

Для пояснения структуры документа и характера языка приведем текст элементарной программы на PostScript с комментариями и результатом ее работы (рис. 20.2):

```
%%BoundingBox: 0 0 612 792
%%Pages: 1
%%Title: Example
%%CreationDate: Wed Mar 29 01:07:49 2000
/Times-Roman findfont           % поиск шрифта
45 scalefont                                  % установка размера шрифта
setfont                                       % назначение шрифта
50 500 moveto                                 % установка текущей координаты
(PostScript example) show       % печать текста
15 250 moveto
450 500 lineto                                % провести линию
stroke                              % прорисовать
4 setlinewidth                      % установить толщину линии
10 10 moveto                                      % рисование рамки
10 550 lineto
500 550 lineto
500 10 lineto
10 10 lineto
stroke
300 220 150 0 360 arc           % провести дугу
0.8 setgray                         % установить цвет заполнителя
fill                                % заполнить сектор
stroke
15 rotate                                             % поворот на 15 градусов
0 setgray
50 50 moveto
/Helvetica-BoldOblique findfont 65 scalefont setfont
(Adobe System) show
showpage                            % показать страницу
```

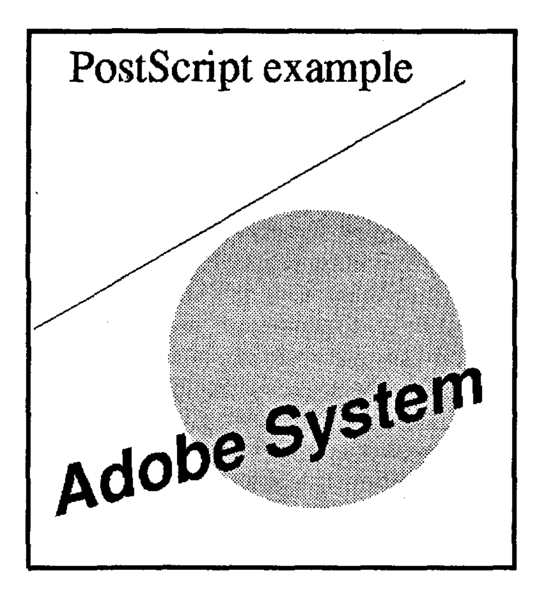


Рис. 20.2. Результат интерпретации PostScript-программы

Как просмотреть и распечатать PS-файлы

При наличии принтера со встроенным интерпретатором языка PostScript печать файлов не составляет никаких трудностей: достаточно отправить PostScript-файл на печать, а сам принтер преобразует команды языка и напечатает документ с максимальным для принтера качеством.

Однако далеко не во всех принтерах встроен интерпретатор языка PostScript, и возникает естественный вопрос: как просмотреть и напечатать файл. Эта проблема стала легко решаться после появления замечательного интерпретатора PostScript для персонального компьютера — пакета Ghostscript. С его помощью можно просматривать PS-файлы на экране, печатать их на принтере, который сам не понимает PostScript, можно исполнять программы в диалоговом режиме. Для работы в среде Windows удобно воспользоваться программой предварительного просмотра ghostview (GSview), которую можно найти на сайте, где лежит дистрибутив Ghostscript (http://www.ghostscript.com). Обе эти программы являются бесплатными и свободно распространяются через Интернет. Установка их на компьютере также проста: надо запустить соответствующую инсталляционную программу (для последней версии это самораспаковывающиеся архивы gsv36w32.exe и gs650w32.exe). Выбор ответов по умолчанию на все задаваемые вопросы даст нужный результат. Инсталляционные программы одновременно устанавливают и необходимые для работы пакета MiKTeX PostScript-шрифты.

При первом обращении к программе gsview32.exe требуется ответить еще на ряд дополнительных вопросов, например о том, какой язык следует использовать в диалоговых окнах программы. Установленные программы являются довольно мощным средством работы с файлами в форматах PostScript и PDF. На рис. 20.3 дано

окно программы GSView с открытым файлом нашего примера. Коротко охарактеризуем пункты меню этой программы. Назначение первого пункта File традиционно — открытие, закрытие и сохранение файлов, печать и выход из программы; кроме того, преобразование PS в EPS-формат. Редактирование Edit включает стандартные возможности Windows для выделения и копирования объектов в буфер обмена, преобразование выделенного в графический формат bmp, сохранение выделенного в файл, поиск в документе. Очень удобным инструментом этого пункта меню является возможность выделения текста из PS и PDF-документа.

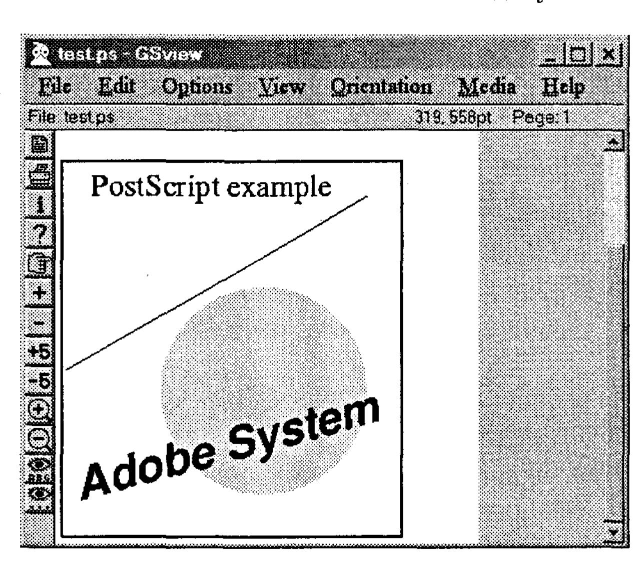


Рис. 20.3. Окно программы GSview

Для установки различных режимов и характеристик просмотра, интерпретации и преобразования документа служит пункт меню Option. Средства для просмотра документа собраны в пункте View, а для управления расположением и размером бумаги предназначены соответственно пункты Orientation и Media. Последний пункт Help является справочником по работе с программой. Для часто используемых команд выделены кнопки в левой части окна программы.

Отметим, что при печати из GSview нужно обратить внимание на правильную установку типа принтера. Просмотр и печать PostScript-файлов поддерживают и некоторые графические редакторы (например, Adobe Photoshop).

Как создать PS-версию документа

Часто возникает необходимость создать PostScript-версию статьи, например для размещения в Интернете, или преобразовать графический объект в EPS-формат. Как уже отмечалось, пакеты Maple и MATLAB позволяют записывать создаваемые графические результаты в формате PostScript. Преобразовать рисунок в PostScript из другого графического формата можно при помощи некоторых графических редакторов (например, Adobe Photoshop). Не существует особых проблем и для преобразования документа LaTeX в PostScript и PDF, для чего существуют

специальные программы dvips.exe и dvipdfm.exe. Эти программы входят в состав рассмотренного выше пакета MikTeX и преобразуют dvi-файл в перечисленные форматы. При работе в редакторе WinEdt для преобразования достаточно нажать на соответствующую кнопку.

Существует универсальный способ преобразования любого документа, подготовленного программами, работающими в операционной системе Windows, в PostScript-формат. Этот способ основан на использовании специального драйвера принтера, изготовленного фирмой Adobe. Эти бесплатные драйверы имеют название AdobePS, плюс номер версии, и их можно найти в Интернете по адресу http://www.adobe.com.

После получения драйвера его нужно установить, запустив соответствующую программу. Вместе с драйвером необходимо иметь и файлы с расширением PPD, которые описывают специфику используемого принтера. Универсальный описатель имеет имя DefaultPostScriptPrinter, и его можно использовать для преобразования в стандарт языка PostScript. В процессе инсталляционного диалога необходимо ответить на ряд вопросов, на которых мы коротко остановимся.

- В качестве типа установки следует выбрать Install a new PostScript Printer.
- В качестве типа соединения следует выбрать It is directly connected to your computer (Local Printer).
- Программе установки необходимо указать директорию, где находится PPD-файл. Для этого нужно нажать кнопку Browse и указать нужный каталог.
- В качестве порта печати следует выбрать FILE: Creates a file on disk (печать в файл).
- Определить имя принтера, по умолчанию предлагается имя, заданное в PPD-файле.
- Ответить на вопрос о печати тестовой страницы. В случае утвердительного ответа на диске будет создан PostScript-файл, содержащий стандартную тестовую страницу печати.
- Определить параметры принтера. Это можно сделать непосредственно при установке драйвера, а можно определить их и позже.

В результате в операционной системе будет определен новый принтер, который нужно указывать при печати документа для преобразования его в формат PostScript. На рис. 20.4 приведено окно свойств установленного принтера. Рассмотрим пункты установки параметров принтера и возможные их значения.

- Пункт Сведения позволяет устанавливать порт принтера (в нашем случае это файл) и используемый драйвер.
- Пункт Общие описывает принтер и позволяет задать страницу, которая будет всегда выводиться на печать, а также определить комментарии, которые будут присутствовать в имени принтера.
- В пункте Paper устанавливается размер листа бумаги, ориентация страницы, количество печатаемых экземпляров, способ подачи бумаги и поля печати.
- Пункт Graphics. Здесь можно установить разрешение принтера, управлять качеством распечатки изображений и текста, настроить насыщенность тона.

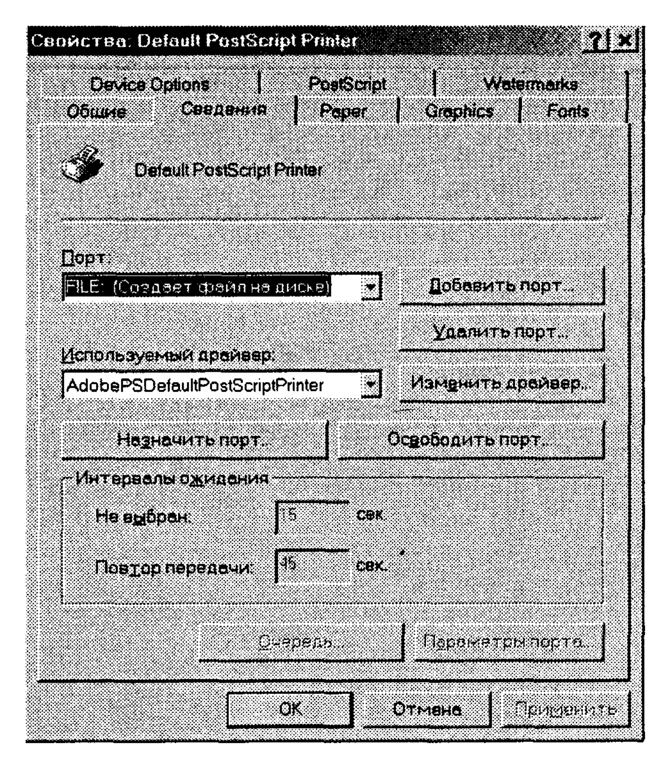


Рис. 20.4. Окно свойств PostScript-принтера

- Пункт Fonts позволяет определить шрифты, которые будут использоваться при печати документа. Можно указать использование PostScript-шрифтов в любом случае, задать таблицу соответствия PostScript и TrueType шрифтов либо определить, чтобы при печати драйвер всегда подставлял системные шрифты TrueType. В последнем случае печать будет происходить медленнее, но все шрифты будут внедрены в PostScript-файл, и полученный документ будет одинаково выглядеть на всех машинах. При распечатке документа, оформленного шрифтами TrueType, на PostScript-принтере происходит преобразование шрифтов в кривые. Это особенно удобно для печати документов на русском языке на нерусифицированных компьютерах. Мы рекомендуем использовать именно такую установку этого параметра, см. рис. 20.5.
- В пункте Device Options можно определить размер используемой принтером памяти.
- Пункт PostScript предназначен для определения параметров PostScript-документа (оптимизация по размеру, по скорости печати и др.).
- Пункт Whatermarks позволяет определить текст, который будет выводиться на всех страницах в качестве фона.

После установки драйвера и определения его параметров можно преобразовать в PostScript-формат любой документ, который можно распечатать в операционной системе Windows, указав в качестве печатающего устройств PostScript-принтер.

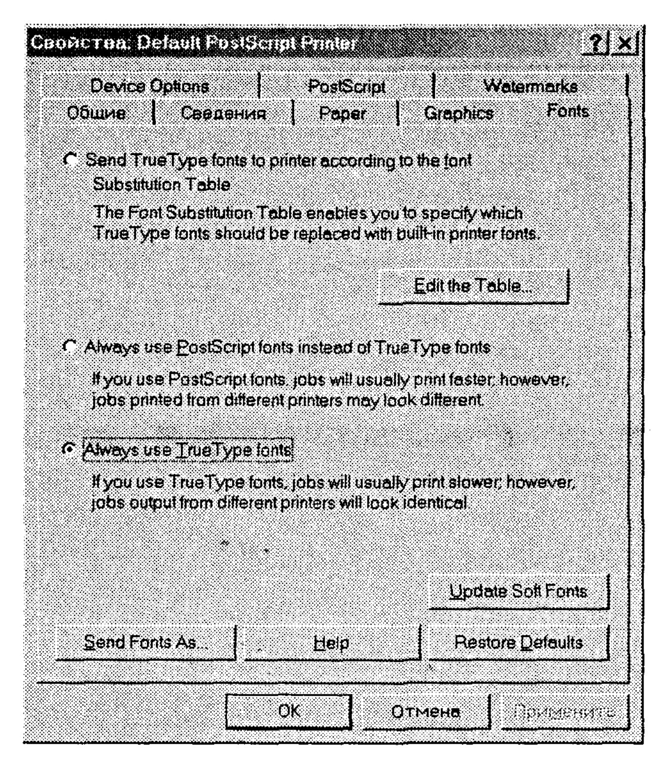


Рис. 20.5. Установка параметров использования шрифтов драйвером PostScript-принтера

Формат PDF и программа Adobe Acrobat Reader

В последнее время все шире распространяется формат PDF (Portable Document Format), который является в некотором смысле развитием формата PostScript. PDF-файлы создаются, как правило, через конвертацию из PostScript-файлов. PDF предложен фирмой Adobe как независимый от платформы формат для создания электронной документации, презентаций, передачи полиграфических материалов и графики через Интернет. Этот формат создавался как компактный способ хранения электронной документации, а потому все данные в нем могут уплотняться, причем к информации разного типа применяются разные, наиболее подходящие для них типы сжатия, например, для включенной в PDF-документ растровой графики обычно используется JPEG-формат.

Многостраничные PDF-файлы могут создаваться программами InDesign, FreeHand, PDFWriter, Acrobat Distiller и некоторыми другими. Для преобразования документов LaTeX в этот формат существует специальная программа dvipdfm.exe, которая входит в состав пакета MikTeX. Для просмотра PDF-файлов может применяться описанная выше программа GSView, но предпочтительнее пользоваться специальной программой Adobe Acrobat Reader. Программа разрабатывается

фирмой Adobe и распространяется бесплатно (см. http://www.adobe. com), хорошо взаимодействует с различными приложениями в операционной системе Windows. Для установки нужно запустить инсталляционную программу и ответить на ряд вопросов, причем можно смело согласиться с установками по умолчанию.

Работа с этой программой проста и не требует подробных инструкций. На рис. 20.6 приведено окно программы с документом, полученным преобразованием нашего первого примера из описания работы с LaTeX в PDF-формат. Содержание пунктов меню и значков очень похоже на аналогичные программы GSView. Последняя версия Adobe Acrobat Reader 4.0 выгодно отличается от более ранних версий тем, что позволяет вырезать текст из PDF-файлов в обычном текстовом формате.

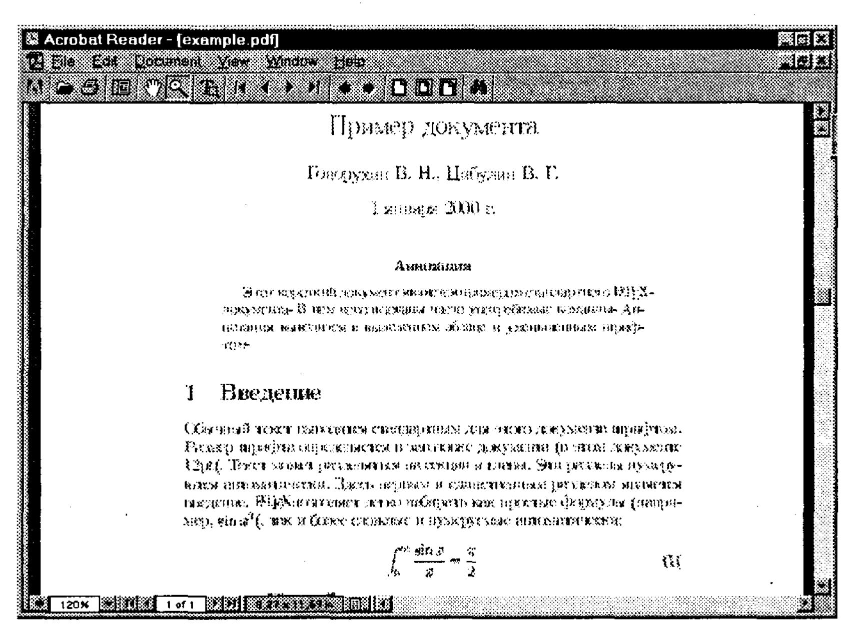


Рис. 20.6. Окно программы Adobe Acrobat Reader 4.0

MS Word и математические тексты

Описанию редактора Word и его возможностей для набора и представления документов посвящены многие отдельные издания (см., например [34]), а потому в этой главе мы ограничимся только рассмотрением набора формул в Word, что отличает именно математические тексты. Коротко будут описаны три возможности набора формул: использование средств форматирования текста самого редактора, применение редактора формул Microsoft Equation и специализированной программы MathType.

Для простых формул, которые не требуют использования знаков интегралов, сумм с пределами, сложных дробей и т. п., можно ограничиться средствами самого Word.

Для набора нижних и верхних индексов необходимо выделить нужный текст и обратиться к пункту меню Формат ▸ Шрифт. Для вставки в формулу греческих и различных специальных символов следует обратиться к пункту меню Вставка ▸ Символ и выбрать нужный шрифт и символ. В качестве пробела при наборе формул следует использовать так называемый неразрывный пробел, который можно найти в меню Вставка ▸ Символ, или воспользоваться специальным сочетанием клавиш, нажав одновременно Ctrl+Shift+Пробел. Приведем пример формулы, набранной описанным способом: $\sin(\pi\ xi)^2$.

Для набора сложных формул существует редактор уравнений Microsoft Equation, который входит в Microsoft Office. Отметим, что MS Equation не всегда устанавливается по умолчанию, и о его наличии надо позаботиться при инсталляции редактора Word, а затем для удобства использования добавить в меню редактора Word соответствующую кнопку. После этого для набора формул достаточно нажать на кнопку или обратиться к пункту меню Вставка ▸ Объект ▸ Microsoft Equation. Можно вызывать редактор формул и в автономном режиме, но работу с формулами при таком вызове мы здесь не описываем.

После вызова редактора формул появляется его меню, панель инструментов (кнопки для групп символов), а в документе возникает пунктирный прямоугольник, в котором размещается формула. Окно редактора формул, запущенного в автономном режиме, дано на рис. 20.7. С помощью пунктов меню можно управлять стилем набираемых символов, их размером. Каждая из девятнадцати кнопок символизирует группу символов: скобки разных видов, шаблоны дробей и индексов, логические символы, интегралы, специальные математические знаки, греческие буквы и т. д. Работа с MS Equation заключается в том, что пользователь набирает с клавиатуры стандартные символы, и выбирает нужные специальные знаки или шаблоны из групп обозначенных кнопками. Опишем последовательность действий для набора формулы, изображенной на рис. 20.7. Сначала набирается последовательность символов sin, затем из левой нижней группы символов выбирается пара скобок, причем после этого внутри скобок возникает поле для ввода. Поместив курсор в это поле, следует выбрать шаблон для большой дроби из второй слева нижней группы символов. Затем курсор устанавливается в числителе дроби и выбирается символ «π» из соответствующей группы, набирается буква «*x*», а затем курсор перемещается в знаменатель и набирается «*a*». Для перемещения курсора можно пользоваться мышью или клавишами стрелок на клавиатуре.

Размер символов можно изменять, используя пункты меню. Отметим, что для ввода некоторых символов предусмотрены комбинации клавиш, например, «→» набирается в результате нажатия Ctrl+K A, а «I» — в результате Ctrl+K E. Символы из любой верхней группы (номера 1-9) можно набирать по номеру группы и символа, для чего следует нажать клавиши Shift+Ctrl+K, ввести номер группы, номер символа и нажать Enter. Например, символ бесконечности возникнет после следующих нажатий клавиш: Shift+Ctrl+K 8 3 Enter. Для некоторых шаблонов также предусмотрены комбинации клавиш, так паре круглых скобок соответствует Ctrl+(, вертикальной дроби — Ctrl+F, и т. д. Пробелы различной величины набираются при помощи следующих комбинаций клавиш (перечислены в порядке увеличения размера): Shift+ Пробел, Alt+Ctrl+Пробел, Ctrl+Пробел, Shift+Ctrl+Пробел.

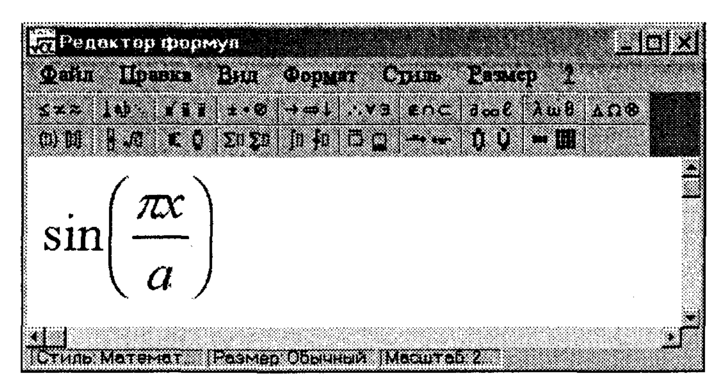


Рис. 20.7. Окно редактора формул Microsoft Equation Editor

Для завершения редактирования формулы достаточно щелкнуть мышкой за ее пределами или нажать клавишу ESC, после чего продолжится работа с Word. Если нужно отредактировать уже существующую формулу, то для этого достаточно дважды щелкнуть мышкой на ней.

Существует еще один способ ввода формул. Для этого следует обратиться к меню Word ▸ Вставка ▸ Поле и в появившейся панели диалога в левом списке выбрать Формула, а в правом — EQ. В возникшем поле ввода можно набирать формулы непосредственно на языке редактора формул. Например, после ввода `EQ \F(sin(x);x)` появится формула: $\frac{\sin(x)}{x}$. Такой способ хоть и редко используется, но позволяет расширить возможности редактора формул, например вставлять графические и другие объекты в формулы, использовать цвет и др. Подробнее о работе с редактором формул, его языке и возможностях можно посмотреть в справке Help.

Более мощным инструментом для набора формул является продукт компании Design Science — редактор MathType, см. http://www.mathtype.com. У этой коммерческой программы есть версии для Windows и Macintosh, MathType может подключаться к известным текстовым редакторам (в том числе Microsoft Word, Corel WordPerfect, AppleWorks) как дополнение, предназначенное для набора математических формул. MathType по сравнению с MS Equation Editor включает сотни дополнительных математических и технических символов и шаблонов, поддержку цвета для профессиональной полиграфии, трансляторы в TeX, LaTeX и MathML-форматы, сохранение формул в виде PostScript, Windows Metafile и GIF-файлах. Работа в этом редакторе аналогична работе с программой MS Equation Editor, но дополнительные сервисные возможности делают ее еще удобнее. Например, можно самостоятельно организовывать собственные символы и выражения и определять для них сочетания клавиш.

После установки в Windows и при наличии редактора MS Word, MathType добавляет свою панель инструментов в главном меню Word (см. рис. 20.8). Кроме уже перечисленных возможностей программа реализует следующие полезные функции: автоматическое изменение расстояний, стилей шрифта и размеров всех формул в документах Word без индивидуального открытия каждой формулы; нумерацию и перекрестные ссылки уравнений (двойной щелчок на ссылке переносит указатель в ту формулу, на которую идет ссылка) и многое другое. О сервисных

и других возможностях программы можно судить по рис. 20.8, на котором приведено окно редактора MathType.

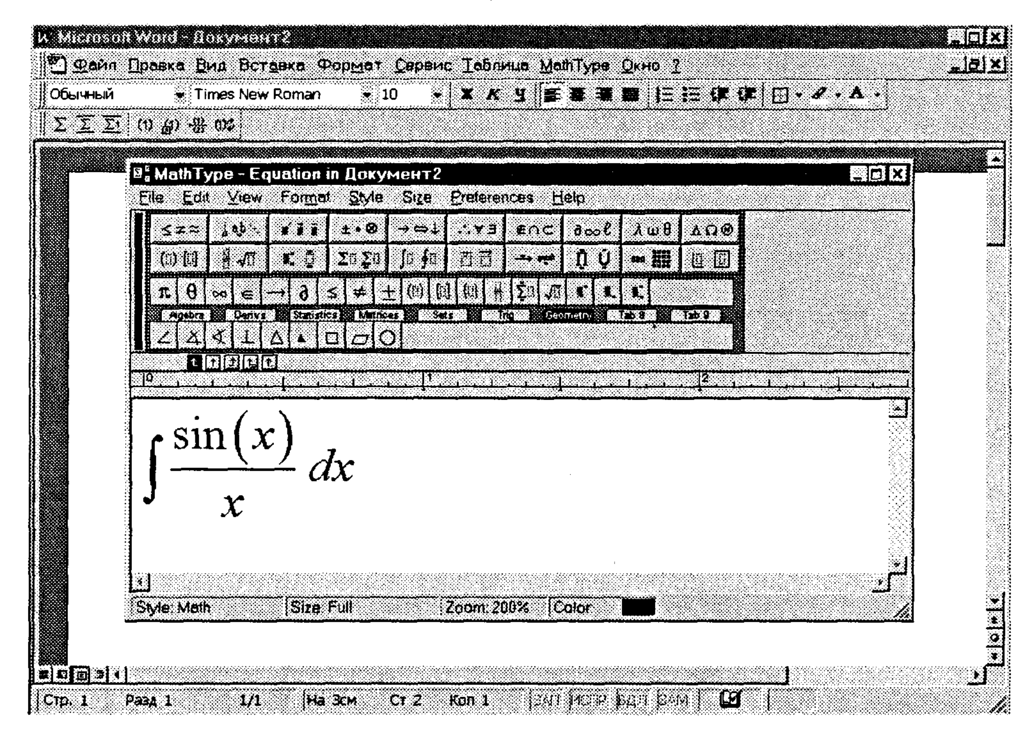


Рис.20.8. Окно программы Microsoft Word с открытым окном редактора формул MathType

Конверторы

Здесь мы коротко остановимся на программах преобразования математических документов из трех наиболее распространенных для публикаций форматов — MS Word, HTML и LaTeX. Мы ограничимся только некоторыми опробованными нами программами, и начнем с конвертации из Word в LaTeX.

Для начинающих пользователей системы LaTeX часто трудным является язык формул, что заставляет их отказаться от ее применения и пользоваться редактором формул MS Equation Editor, так как редактирование формул в нем наглядно и относительно просто. В предыдущем разделе была описана программа MathType для набора формул, которая понимает язык Equation Editor и может преобразовывать формулы в синтаксис LaTeX. Однако эта программа является коммерческой и не всегда доступна. Фирмой Design Science бесплатно распространяется через Интернет (http://www.mathtype.com) программа TeXaide, которая внешне похожа на редактор формул Microsoft Equation Editor, полностью понимает его язык и включает возможность преобразования набранной формулы в различные варианты TeX. Для установки достаточно получить по сети инсталляционную программу и запустить ее. Работа с программой полностью идентична работе с редактором формул

MS Equation Editor, см. предыдущий раздел. В меню программы имеется пункт Edit ▸ Translators, где можно выбрать версию TeX, для которой будет переводиться формула. Для преобразования формулы достаточно ее выделить, обратиться к пункту меню Edit и скопировать выделенное в буфер Windows (пункт Copy). После этого в текстовом редакторе, используемом для редактирования LaTeX, например WinEdt, достаточно вставить содержимое буфера (обычно пункт Edit ▸ Paste). На рис. 20.9 дано окно программы TeXaide с набранной формулой, а ниже приводится результат ее преобразования:

```
\[  \int {  \frac{{\sin \left( x \right)}}{x}} \;dx \]
```

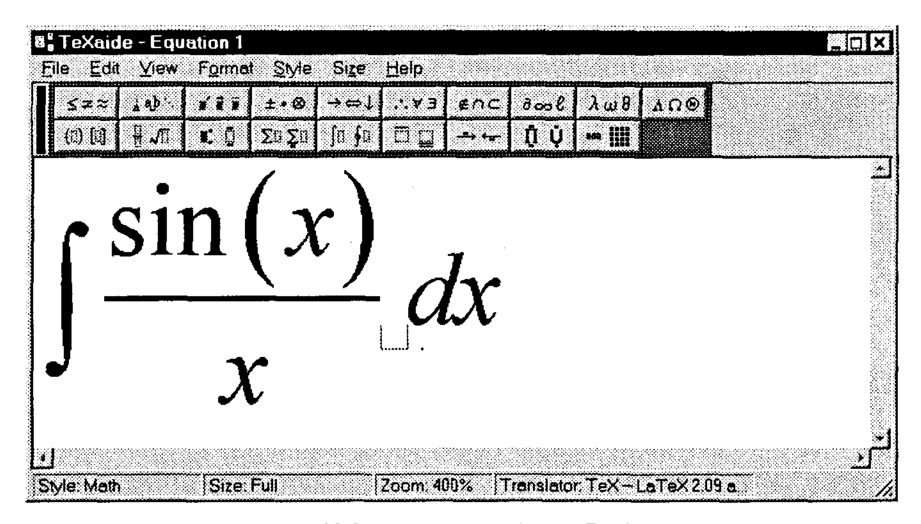


Рис. 20.9. Окно редактора формул TeXaide

Преобразовывать весь документ из формата Word в LaTeX позволяет программа Word2TeX. Это условно-бесплатная программа, которая свободно распространяется, но для незарегистрированной версии часть возможностей отключена. Для установки Word2TeX необходимо получить через Интернет (http://www.word2tex.com) инсталляционный самораспаковывающийся архив, а затем запустить программу word2tex.exe. В результате инсталляции в меню редактора Word появится новый пункт Word2TeX, который предназначен для установки параметров перевода текста из одного формата в другой и получения информации. На рис. 20.10 приведено окно редактора Word с модифицированным меню и открытым диалогом установки параметров транслятора. Для конвертации документа Word в LaTeX достаточно обратиться к пункту Файл ▸ Сохранить как меню редактора и выбрать из предложенного списка типов файлов tex. В результате в указанном пользователем каталоге на диске будет создан LaTeX-документ. При преобразовании успешно переводятся не только текст с элементами форматирования, но и формулы. Отметим, что программа ориентирована на конвертацию текстов на английском языке, а потому после преобразования для документов на иных языках нужно вставить необходимые строки для их поддержки. Далее приведен результат конвертации небольшого документа (см. рис. 20.10) и добавлены две строки подключения символов кириллицы для пакета MikTeX:

```
% Converted from Microsoft Word to LaTeX
% by Chikrii SoftLab Word2TeX converter (version 2.0)
% Copyright(C) 1999-2001 Kirill A. Chikrii, Anna V. Chikrii
% All rights reserved. http://www.word2tex.com/
%Warning: You are using UNREGISTERED Word2TeX!
\documentclass [12pt]{article}
% Следующая строка добавлена авторами книги
\usepackage[cp1251]{inputenc} \usepackage[russian]{babel}
\begin{document}
\section{Пример конвертации из Word в LaTeX}
Программа Word2TeX преобразует документы в формате Word в LaTeX. Преобразуется не только
текст вместе с форматированием, но и формулы. \textit{Например:} $F\left( {x} \right) =
{\sum\limits_{i = 1}^{n} {\int {\pi \sin \left( {x} \right)dx}} }$
\end{document}
```

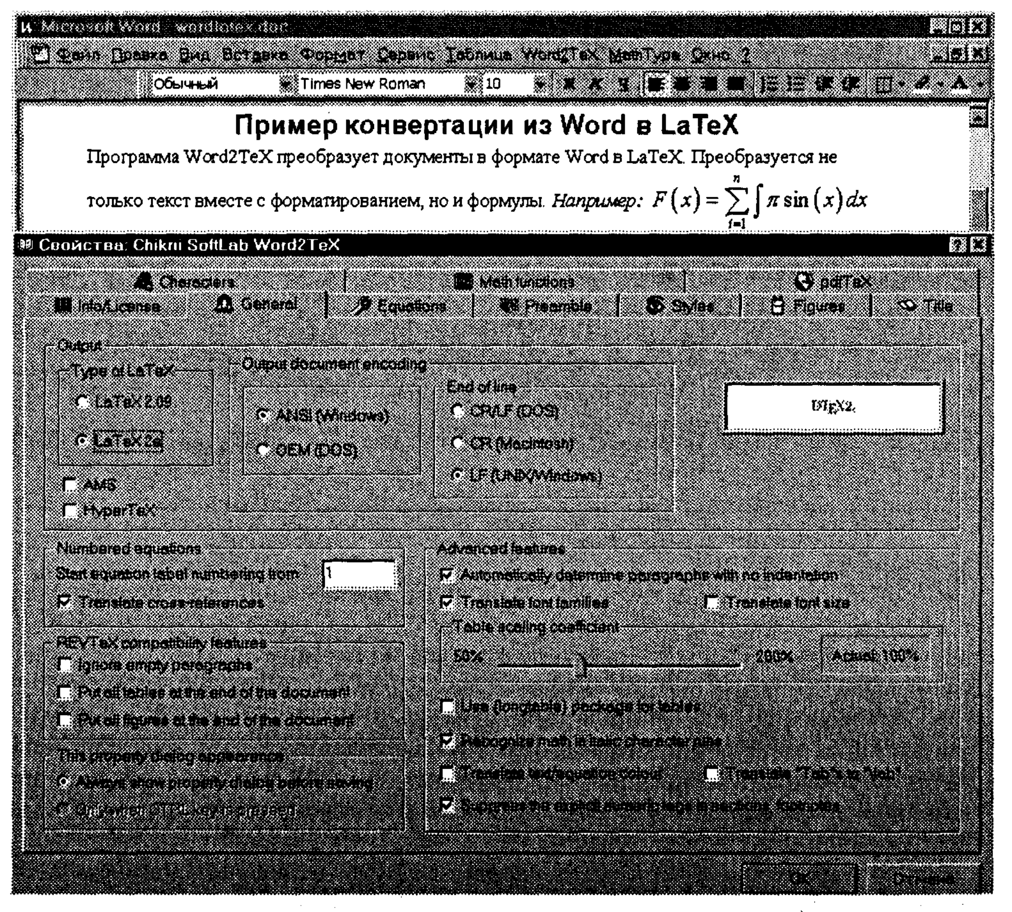


Рис. 20.10. Окно редактора Word после установки конвертора Word2TeX и диалогом установки параметров перевода

Таким образом, с переводом из формата Word в формат LaTeX особых трудностей не возникает. Однако для эффективной конвертации из LaTeX в Word нам не удалось найти удовлетворительных инструментов. Существует несколько программ преобразования из LaTeX в формат RTF, который поддерживается редактором Word, но они не преобразуют формулы в синтаксис MS Equation Editor, а в лучшем случае представляют их в виде графических объектов. Следует упомянуть еще коммерческую программу TeXport, которая, судя по описанию (см. http://www.ktalk.com), может преобразовывать и формулы в форматы различных текстовых редакторов. Еще одним, хотя и не совсем приемлемым, способом решения этой задачи является преобразование LaTeX-документа в формат HTML, который также поддерживается последними версиями Word.

Для преобразования TeX-документов в язык HTML существует довольно много программ, но мы остановимся только на двух. Одна из них, TTH, входит в последние версии пакета MikTeX и автоматически подключается к редактору WinEdt. При работе в WinEdt для преобразования достаточно обратиться к соответствующему пункту меню. Бесплатная программа TTH (http://hutchinson.belmont.ma.us/tth) использует язык HTML не только для форматирования текста, но и представления формул. При переводе сохраняется нумерация формул, ссылки и другие средства LaTeX. На рис. 20.11 дан результат преобразования программой TTH документа LaTeX приведенного выше в этом разделе в окне редактора Word. Видно, что формулы преобразовались не вполне удовлетворительно.

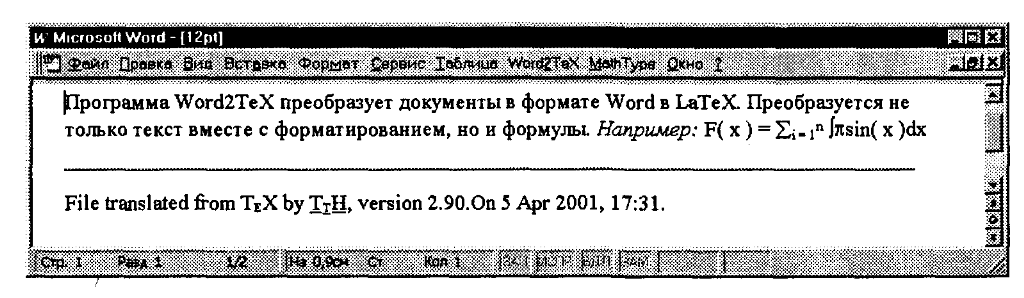


Рис. 20.11. Результат преобразования документа LaTeX в формат HTML

Еще одним средством преобразования документов, написанных с помощью LaTeX, в HTML-формат является программа LaTeX2HTML. Ее можно легко найти в Интернете, например в CTAN (см. главу 21 раздел «TeX в Интернете»). Эта программа поддерживает практически все возможности как LaTex (новые команды, счетчики и другие), так и HTML (графика, списки, таблицы стилей и др.). Формулы она преобразует в рисунки в формате GIF. Но при работе с ней существуют и трудности, например, не всегда корректно выполняются преобразования, поэтому полученный HTML-документ часто требует ручной правки.

Графические системы и файлы

В этом разделе мы коротко опишем основные форматы графических файлов. Эта информация часто бывает полезной при изготовлении рисунков, изображающих

математические результаты. Рассматриваемые в этой книге математические пакеты позволяют сохранять рисунки в различных форматах, но часто возникает необходимость их корректировки при помощи графических редакторов. Кроме того, знание форматов файлов и их возможностей является важным фактором в подготовке изданий, изображений для Интернета и в компьютерной графике вообще.

Мы не будем здесь описывать графические редакторы. В изложении книги мы ориентируемся на то, что в основном наши читатели используют в своей работе операционную систему Windows (95, NT, 98, ...). В состав этой операционной системы входят графические редакторы Paint и Microsoft Photo Editor. Мощным графическим пакетом, работающим в среде Windows, является пакет Adobe Photoshop, который поддерживает большинство графических форматов. В электронной библиотеке издательства «Питер-пресс» (http://www.piter-press.ru/) находится полная электронная версия книги [31], описывающая работу с этим пакетом. Об этих программных продуктах существует много литературы (см. например, [29, 31, 35] и другие), да и работа в них не должна вызывать трудностей в силу удобного и прозрачного интерфейса. Перечисленные программы могут преобразовывать одни графические форматы в другие. Ниже мы коротко перечислим основные графические форматы.

Все графические данные в компьютере можно разделить на две группы: растровую и векторную графику. Векторы являются математическим описанием объектов в физических координатах относительно точки начала координат. Например, чтобы компьютер нарисовал прямую, нужны координаты двух точек, для дуги задается радиус и т. д. Преимущества таких картинок — меньший размер по сравнению с растровыми (подробнее — далее) и возможность масштабирования без потери качества (все размеры и координаты линий изменяются пропорционально, и результат зависит лишь от возможностей устройства вывода). Примерами векторных графических объектов могут служить графические структуры Maple (см. главу 6 «Графика Maple») и дескрипторная графика MATLAB (см. главу 14 «Графика MATLAB»).

Растровые изображения представляют собой набор точек или пикселов («битовую карту» — bitmap). Каждая точка изображения имеет параметры — цвет, прозрачность и др. Такие форматы используются там, где важна передача полутонов, цветовых переходов. Если геометрические размеры картинки большие, то набор описаний каждой точки (то есть весь файл целиком) может быть очень большой. Для растровых изображений существует еще один важный параметр — разрешение. Разрешение — это число точек (пикселов) на единицу длины. Эта величина обычно измеряется в точках/дюйм: dpi (dots-per-inch) или ppi (pixels-per-inch). Разрешение — один из определяющих факторов качества изображения.

Способы представления цвета

Здесь речь пойдет о методах кодирования цвета в компьютере. Считается, что человек воспринимает цветовую информацию с помощью рецепторов, которые можно разделить на три группы, каждая из которых воспринимает только один цвет — красный, зеленый или синий. Другие цвета формируются комбинациями этих трех основных красок в различных пропорциях. Цвета также характеризуются различимостью участков, которую называют светлотой, и насыщенностью, определяю-

щей степень отличия данного цвета от чистого основного цветового тона. Различают ахроматические цвета (белый, серый и черный), которые отличаются только светлотой, и хроматические цвета, имеющие насыщенность, светлоту и цветовой тон.

На трех цветах основан один из методов кодирования цветовой информации, так называемая RGB-модель. Каждый цвет в этой цветовой модели описывается пропорциями красного (Red), зеленого (Green) и синего (Blue) цветов. Эта цветовая модель широко используется во многих компьютерных программах, значение доли каждого цвета часто задается целым числом из интервала 0-255 или 0-63, а иногда числом с плавающей запятой из отрезка [0,1]. Последний вариант используется в Maple и MATLAB. Соответствующие коды некоторых основных цветов перечислены в таблице 14.11 второй части данной книги. В основном цветовая RGB-модель используется при создании и обработке компьютерной графики для изображения на экране монитора.

При подготовке публикаций к печати обычно используется цветовая модель CMYK. Базовыми здесь являются цвета, полученные вычитанием основных цветов модели RGB из белого: голубой (Cyan) = белый — красный, пурпурный (Magenta) = белый — зеленый и желтый (Yellow) = белый — синий. Однако сочетанием этих цветов не удается получить чистый черный (blacK) цвет, и потому он дополнительно добавлен в цветовую модель. Такое кодирование цветов удобно для проведения цветоделения, которое является обязательным этапом для печати цветных изображений на полиграфическом оборудовании.

Широко распространена и цветовая модель HSB, которая построена на описании цвета оттенком (Hue), насыщенностью (Saturation) и яркостью (Brightness). Эту модель обычно используют для имитации приемов и инструментов художника при работе на компьютере. В некоторых программах существуют упрощенные схемы описания цвета на основе HSB, так в Maple кроме задания цвета с помощью RGB возможно его определение оттенком (Hue).

Понятно, что количество цветов в компьютерном образе рисунка зависит от объема выделяемой под хранение этой информации памяти. Если для кодирования цвета выделен один бит (может принимать значения 0 или 1), то компьютерный рисунок будет черно-белым, если один восьмибитовый байт — то доступно 256 цветов и, т. д. Присутствующие в рисунке цвета обычно содержатся в так называемой палитре, состоящей из ячеек, в каждой из которых хранится код цвета в одной из цветовых моделей. В зависимости от выделяемой памяти палитра может содержать 2, 16, 256, 65 536 и более цветов. При создании рисунков следует помнить, что количество цветовой информации, сохраненной в файле рисунка, сильно влияет на его размер.

Форматы графических файлов

Форматы графических файлов — это способ представления графики в памяти компьютера. Некоторые форматы поддерживают только векторные или только растровые изображения; в то же время многие форматы позволяют использовать оба типа данных в одном файле. Приведем наиболее распространенные графические форматы, сопроводив их кратким описанием. Большинство перечисленных форматов поддерживается программами Maple и MATLAB.

- Формат AI (Adobe Illustrator). Этот векторный формат разработан фирмой Adobe System. Широко используется в издательской деятельности.
- Формат BMP. Является стандартом представления растровых изображений для DOS и Windows.
- Формат BMP. Является стандартом представления растровых изображений для DOS и Windows. Он поддерживается всеми графическими редакторами, работающими под управлением этих операционных систем. Способен хранить как индексированный (до 256 цветов), так и RGB-цвет (16 700 000 оттенков). Область применения его узка — он не используется ни в Интернете, ни для качественной печати.
- Формат EPS (Encapsulated PostScript). Фактически это один из вариантов PostScript-формата. Подробнее о нем см. в разделе «Создание и использование PostScript-файлов».
- Формат GIF (Graphics Interchange Format) обычно используется для представления индексированных (до 256 цветов) цветных изображений и HTML-документов в сети Интернет. GIF является уплотненным форматом, разработанным с целью ускорения пересылки файлов по сети, и является аппаратно-независимым. Формат создан в 1987 году (GIF87a) фирмой CompuServe, а в 1989-м модифицирован (GIF89a); были добавлены поддержка прозрачности и анимации.
- Формат JPEG (Joint Photographic Experts Group) широко используется для отображения фотографий и других тоновых изображений в Интернете. В JPEG применяется очень эффективный алгоритм уплотнения, который нередко дает значительное сокращение объема файла за счет отбрасывания избыточной информации, практически не влияющей на отображение документа. При открытии JPEG-файла происходит его автоматическая распаковка. Однако следует помнить, что чем выше уровень компрессии, тем больше данных отбрасывается, а значит, тем ниже качество. Формат аппаратно независим, но он относительно нов и не понимается старыми программами (до 1995 года).
- Формат PCX. Растровый формат PCX, разработанный фирмой Z-soft для программы PC Paintbrush, очень широко используется на IBM PC-совместимых компьютерах. Большинство программ для PC поддерживает пятую версию PCX-формата. Формат PCX поддерживает RLE-сжатие файлов. Изображения могут иметь глубину цвета 1, 4, 8 или 24 бита (то есть от черно-белого изображения до картинки с 16 777 216 цветами).
- Формат PDF предложен фирмой Adobe как независимый от платформы формат для создания электронной документации, размещения публикаций в Интернет, передачи верстки и графики через сети.
- Формат PostScript. Универсальный формат векторных файлов. Подробнее о нем см. в разделе «Создание и использование PostScript-файлов».
- Формат PSD (Adobe Photoshop) разработан фирмой Adobe System. Широко используется в коммерческой графике.
- Формат TIFF (Tagged-Image File Format) применяется для обмена документами между различными программами и компьютерными платформами. TIFF

представляет собой гибкий формат растровой графики, который поддерживается практически всеми программами рисования, редактирования изображений и верстки. Практически все сканеры способны генерировать TIFF-файлы.

- Формат WMF (Windows Metafile). Универсальный формат для хранения смешанных (растровых и векторных) изображений в Windows. Поддерживается преимущественно офисными программами и некоторыми графическими редакторами.

Мы перечислили основные графические форматы. Нет рекомендаций на все случаи жизни: какой формат выбрать, зависит от поставленной задачи. От себя мы можем сказать, что для использования графики в LaTeX-документах предпочтительнее использовать Encapsulated PostScript, при размещении статей в виде файлов в Интернете удобны PDF и PostScript, а при создании страниц для Интернета нужны рисунки в форматах JPEG или GIF.

21 ГЛАВА Интернет и математика

Понятно, что в одной главе невозможно достаточно полно описать все математические ресурсы в Интернете, а потому мы ограничимся только наиболее важными, на наш взгляд, и близкими к теме книги. Изложение в основном ориентировано на бесплатные программные средства и ресурсы. Сетевое пространство постоянно развивается, и адреса ресурсов могут меняться, но мы старались использовать ссылки только на давно и постоянно функционирующие сайты и архивы.

Сначала рассмотрены математические программы и библиотеки и их информационная поддержка. Описаны ресурсы, посвященные пакетам Maple, MATLAB и TeX, а также ряд бесплатных пакетов и библиотек. Второй раздел является кратким путеводителем по различным математическим информационным ресурсам в Сети. В последнем разделе главы даны введение в язык HTML и некоторые советы по размещению в Интернете математических публикаций.

Математические программы в Интернете

На данный момент в Интернете имеется огромное количество информационных ресурсов с коммерческими продуктами, бесплатных математических программ, библиотек алгоритмов реализующих основные численные методы на различных языках программирования. Их доступность, охват практически всех направлений компьютерного анализа математических задач позволяют значительно сократить время и трудозатраты как студента, так и квалифицированного исследователя. В этом разделе мы бегло перечислим и опишем некоторые из таких ресурсов в Интернете. Отметим, что значительное их число, к сожалению, на английском языке.

Maple в Интернете

Официальный сайт компании Waterloo Maple Software (http://www.maplesoft.com) содержит множество полезной информации для пользователей программы Maple.

Основная страница содержит ссылки на информацию о самой фирме, о новых продуктах компании, а также предоставляет возможности интерактивной регистрации купленных продуктов. Поддержка работы пользователей включает бесплатное распространение обновлений пакета, список замеченных ошибок и неточностей в работе, ответы на наиболее часто повторяющиеся вопросы. В частности, на странице технической поддержки можно прочитать инструкцию по установке доработанной версии (на момент написания книги — Maple 6.02), а также скачать необходимый инсталляционный файл. В разделе публикаций можно бесплатно получить (файл в формате pdf) любой из выпусков электронного журнала «The Maple Reporter», просмотреть список выпущенных книг о Maple, прочитать статьи о применении пакета, а также обзоры и результаты тестов из других изданий. Раздел Links содержит большое количество ссылок на различные ресурсы Интернета, посвященные пакету. Компания Waterloo Maple Software поддерживает группу новостей и дискуссионный форум. Кроме того, на сайте можно найти адреса для связи с фирмой, просмотреть список вакансий компании.

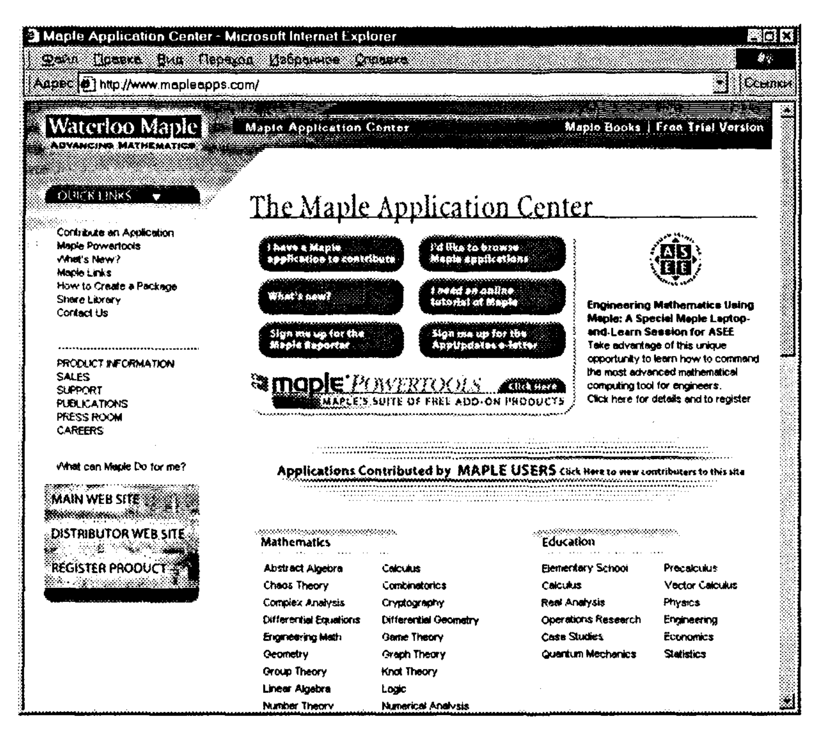


Рис. 21.1. Страница Maple Application Center

Особое место занимает так называемый «центр приложений Maple» (Maple Application Center, http://www.mapleapps.com), содержащий множество программ для решения задач из различных областей средствами Maple. На рис. 21.1 приведена страница этого центра. Она реализована в виде гипертекстовых ссылок и структурирована по различным направлениям образования, техники, математики и других наук. Обращаясь к нужной ссылке, можно получить список имеющихся про-

грамм из этой предметной области, прочитать описание каждой из них, получить текст непосредственно в браузере или в виде Maple-файла. Все программы тщательно проверены, а их авторами являются ученые разных стран и специальностей. Отметим, что любой пользователь пакета может отправить свою программу, и после тестирования и удачного рецензирования она пополнит центр приложений. В разделе Maple Powertools собраны библиотеки для решения некоторых классов задач. Здесь можно найти библиотеки, реализующие метод конечных элементов, методы оптимизации и мн. др.

Теперь остановимся на Maple-ресурсах Интернета на русском языке. В первую очередь рекомендуем сайт фирмы Softline, которая распространяет научное программное обеспечение ведущих фирм в России. По адресу http://www.softline.ru можно найти не только описание последних версий пакета, но и множество другой полезной информации, в частности электронный вариант книги Б. М. Манзона [4]. Еще одним русскоязычным источником большого количества информации о Maple является образовательный математический сайт http://www.exponenta.ru. Здесь помещен электронный вариант книги [6] и указано много полезных ссылок. В заключение приведем адрес страницы, поддерживаемой авторами книги, на которой содержатся полезные ссылки и краткое руководство по Maple: http://www.math.rsu.ru/mexmat/kvm/MME

MATLAB в Интернете

Мощную информационную поддержку своих продуктов осуществляет и компания Mathworks, которая производит пакет MATLAB. На рис. 21.2 приведена главная страница сайта компании (http://www.mathworks.com).

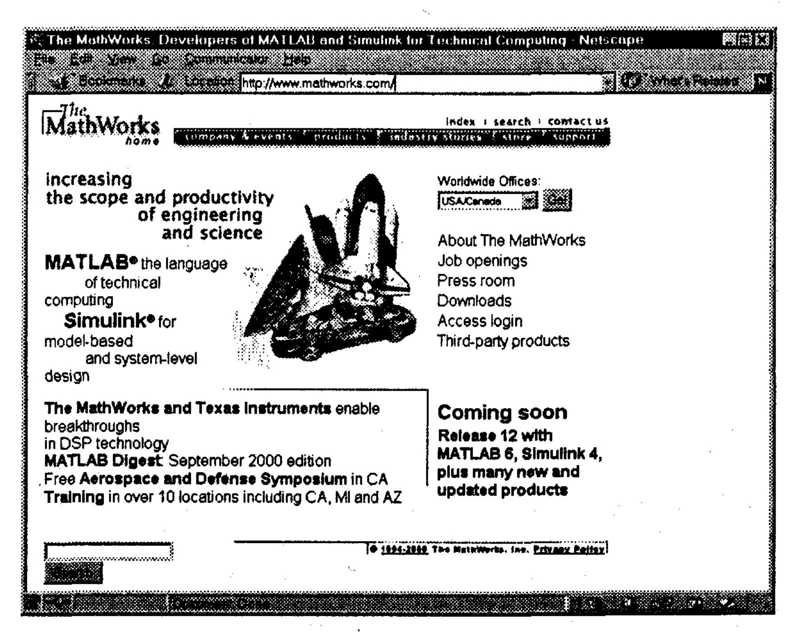


Рис. 21.2. Главная страница сайта компании Mathworks

Помимо рекламы и описаний программных продуктов, выпускаемых фирмой, здесь содержатся ссылки на файлы их модернизаций, новости, объявления о работе и многое другое. Очень полезными, на наш взгляд, являются архивы различных программ для MATLAB, предоставляемых пользователями пакета, большой список полезных ссылок и группа новостей, подключившись к которой можно почерпнуть много важной и постоянно обновляющейся информации.

Фирма Mathworks предоставляет возможность пробной эксплуатации всех своих продуктов, для чего достаточно заполнить анкету и выбрать интересующий продукт. После этого по указанному в анкете электронному адресу будет выслан пароль, который позволит скачать необходимые для установки файлы через Интернет и в течение месяца использовать программу. Любой пользователь может совершенно бесплатно получить в виде PDF-файлов полный набор документации MATLAB и всех существующих коммерческих библиотек (Toolboxes) на английском языке, например книги [21–28]. Для зарегистрированных пользователей становятся доступны различные сервисные возможности, включая даже прослеживание пути высланной в его адрес почтовой бандероли.

Перевод на русский язык краткого курса по MATLAB (файлы в формате PDF), входящего в поставку пакета, и рекламных материалов можно найти на сайте компании Softline (http://www.softline.ru). Здесь же находится ряд статей о применении пакета для решения задач науки и техники, много полезных ссылок и информации на русском языке. Не меньше русскоязычных материалов содержит образовательный математический сайт Exponenta (http://www.exponenta.ru), в частности, здесь помещен электронный вариант книги В. Г. Потемкина «Введение в MATLAB».

TeX в Интернете

Изначально система верстки TeX задумывалась как свободно распространяемая, а потому в Интернете имеется огромное количество информации о ней. Наиболее полным и постоянно обновляющимся является так называемый архив CTAN (Comprehensive TeX Archive Network). Наряду с сайтом http://www.ctan.org, существует большое количество копий этого архива, которые расположены в различных странах, мы приведем только некоторые из них:

- С описанием на английском языке: ftp://ftp.chg.ru/pub/TeX/CTAN/; ftp://ctan.math.utah.edu/; ftp://cis.uniroma2.it/tex.
- С описанием на немецком языке: http://www.dante.de/software/ctan/.

Первый сайт кроме копии архива содержит множество пакетов и информации для русификации различных версий TeX. Чтобы найти ее, нужно подняться на один уровень файлового дерева: ftp://ftp.chg.ru/pub/TeX/. Среди других русскоязычных ресурсов Интернета, посвященных системе TeX и содержащих много полезного, отметим наиболее полные:

- Ассоциация пользователей кириллического TeX'a (http://www.cemi.rssi.ru/cyrtug).
- Кириллический TeX на сервере МГУ (http://tex.msu.ru/).

В данной книге мы рассматривали набор текстов с использованием макропакета LaTeX, который является одним из наиболее распространенных. Все новости о LaTeX можно найти по адресу: http://www.latex-project.org/.

Библиотеки алгоритмов и программ

Несмотря на наличие универсальных математических пакетов, таких как Maple, MATLAB, Mathematica, MathCad, Macsyma и др., программирование на алгоритмических языках по-прежнему широко используется для решения многих задач. При таком компьютерном исследовании очень полезными оказываются библиотеки программ, реализующие основные численные методы и алгоритмы. В этих библиотеках собраны хорошо проверенные процедуры для решения самых разнообразных задач. Мы остановимся только на наиболее полных и универсальных ресурсах.

Начнем с так называемых «Численных рецептов» (Numerical Recipes). По адресу http://www.nr.com размещен электронный вариант одноименной книги, издаваемой Cambridge University Press. Эта книга содержит большинство классических численных методов, включая их описание на английском языке и тексты процедур. Доступны три варианта этой книги: с программами на языке С, Fortran77 и Fortran90. Файлы с главами книги хранятся в формате PDF или PostScript. Используя гиперссылки, пользователь выбирает интересующий его вариант книги и попадает на страницу с ее оглавлением. Нажав на нужную ссылку, можно получить файл с текстом любой главы, прочитать описание метода и извлечь текст процедуры.

Библиотека Netlib (http://www.netlib.org) содержит не только тексты процедур, но и статьи из области вычислительной математики, а также множество полезных ссылок на аналогичные ресурсы. Найти нужную процедуру можно, воспользовавшись средствами поиска по ключевым словам или просмотрев полный список архива. Еще одной большой коллекцией разнообразных численных процедур является архив GAMS (Guide to Available Mathematical Software), расположенный по адресу: http://gams.nist.gov. Он также снабжен средствами поиска, ссылками на литературу и ресурсы Интернета. На решение задач большой сложности (в смысле объемов вычислений), в частности с использованием параллельных компьютеров, ориентирован архив NHSE (http://www.nhse.org). Много различных процедур и полезных ссылок находится на немецком сайте научных вычислений (http://www.scicomp.uni-erlangen.de). Библиотеку алгоритмов международного института ядерной физики можно получить по адресу: http://wwwinfo.cern.ch/asd, а затем установить на своем компьютере. Отметим, что большинство процедур, содержащихся в этих архивах, реализованы на языках FORTRAN и С.

Кроме перечисленных универсальных библиотек алгоритмов существует великое множество специализированных, которые можно найти, воспользовавшись любыми поисковыми системами. Так, по адресу http://www.unige.ch/math/folks/hairer находятся процедуры для решения систем обыкновенных дифференциальных уравнений, описанные в книге [73]. Те, кто программирует на языке Pascal, может найти реализацию многих численных методов по следующей ссылке: http://www.cs.vu.nl/~jprins/tp.php. Для любителей объектно-ориентированного программирования предназначен сайт http://oonumerics.org.

Из ресурсов на русском языке отметим сайт библиотеки численного анализа НИВЦ МГУ (http://www.srcc.msu.su/num_anal), уже упоминавшийся выше проект Exponenta (http://www.exponenta.ru), библиотеку алгоритмов, размещенную по адресу http://alglib.chat.ru/, и сервер параллельных вычислений http://parallel.ru.

Бесплатные математические пакеты

Описанные в книге пакеты Maple и MATLAB обладают большими возможностями и удобством, но являются довольно дорогими, их цена обычно превышает тысячу долларов. Для большинства пользователей стран СНГ, которые не используют пиратские версии, их приобретение практически невозможно. Кроме того, несмотря на универсальность, далеко не все аспекты компьютерного анализа в них отражены достаточно полно. Частично решить эти проблемы позволяют как бесплатные аналоги перечисленных пакетов, так и другие программы, которые разрабатываются во многих университетах и научных центрах. В этом разделе мы перечислим некоторые из них.

Программа MuPAD

Программу MuPAD можно в некотором смысле считать аналогом пакета Maple. Она разрабатывается уже несколько лет в университете Падеборна, Германия. Первое время программа была бесплатной, но последние реализации MuPAD для Windows являются коммерческими, хотя можно получить версию, работающую месяц, а их аналоги для различных версий Linux требуют только регистрации и найти их можно по адресу: http://www.mupad.de.

Работа в среде MuPAD очень похожа на работу с Maple в режиме сессии, то есть пользователь вводит команды и сразу получает результат. Рабочий документ включает области ввода, вывода, графики. Кроме того, и синтаксис языка во многом идентичен синтаксису Maple. На рис. 21.3 дано окно программы с небольшим примером.

Аналоги MATLAB – Octave и Scilab

Сейчас существует сразу два бесплатных аналога пакета MATLAB. Первый из них — Scilab, который развивается французским Институтом исследований в области информатики и автоматизации (INRIA), а второй — проект Octave. Обе программы свободно распространяются через Интернет соответственно по адресам: http://www-rocq.inria.fr/scilab/ и http://www.octave.org/.

Программы имеют интерфейс, аналогичный интерфейсу ранних версий MATLAB, сохраняют идеологию MATLAB, и практически повторяют его язык. Для установки программ достаточно скачать необходимый инсталляционный набор файлов и запустить соответствующую программу. Существуют версии для операционных систем Windows, Linux, FreeBSD и других.

Работа с обеими программами практически полностью аналогична работе с MATLAB. Читателю, уже знакомому со второй частью данной книги, это станет очевидно из просмотра двух рисунков. На рис. 21.4 и 21.5 соответственно представлены окна программы Scilab и Octave с построенным графиком функции синус и набором нужных для этого команд. Однако отметим, что далеко не все возможности MATLAB реализованы в этих бесплатных аналогах, особенно команды последних версий.

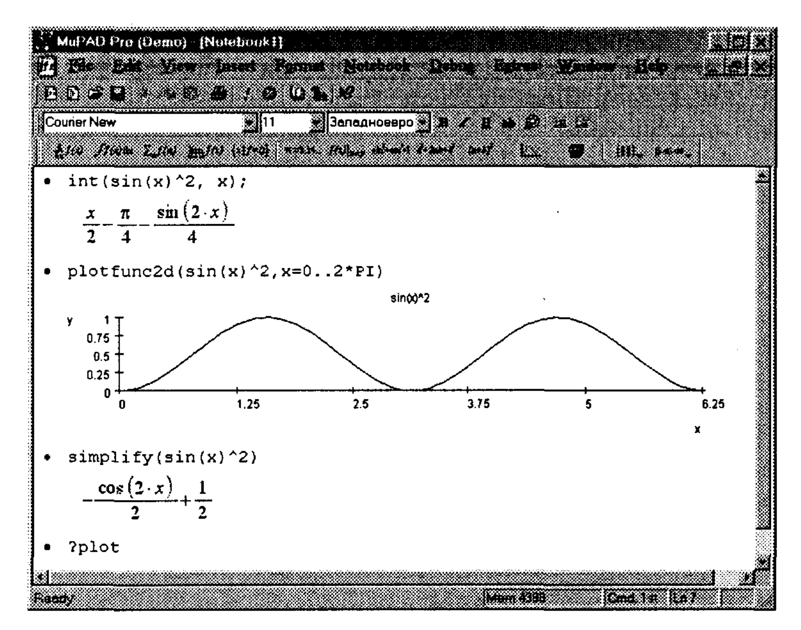


Рис. 21.3. Окно программы MuPAD

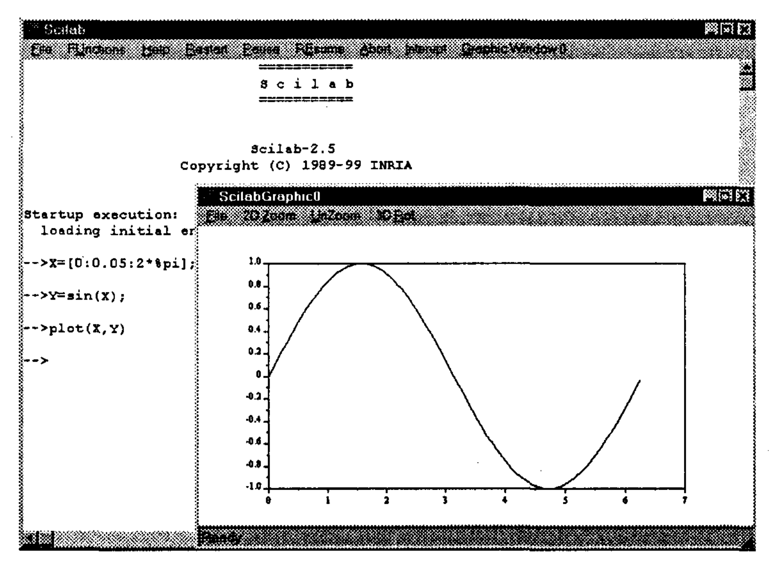


Рис. 21.4. Окна программы Scilab

Пакеты для исследования динамических систем

Многие задачи науки и техники формулируются в виде конечномерных систем дифференциальных и разностных уравнений, причем методы анализа таких систем, по нашему мнению, недостаточно отражены в универсальных системах Maple и MATLAB. Имеется несколько программ, позволяющих проводить исследование

систем разностных и дифференциальных уравнений. Почти все такие программы бесплатны и свободно распространяются через Интернет. Далее мы перечислим некоторые из них, сопровождая изложение короткой аннотацией и ссылкой на соответствующий сетевой ресурс.

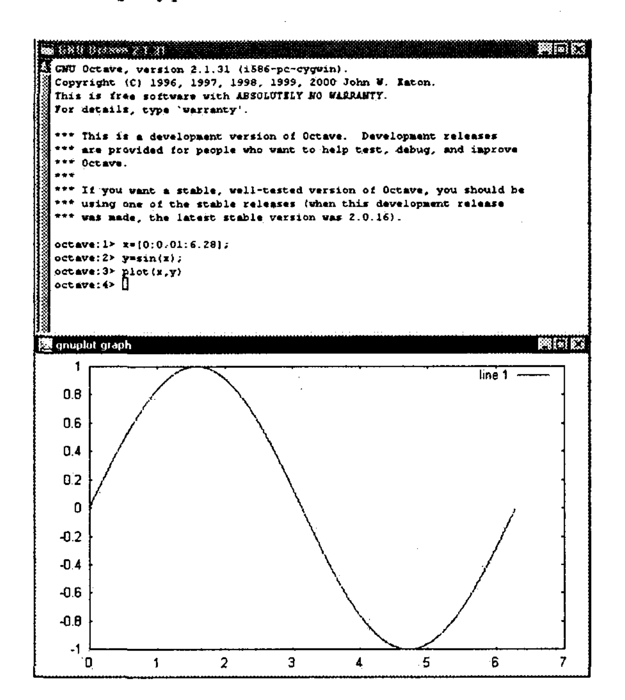


Рис. 21.5. Окна программы Octave

Одной из самых мощных по возможностям исследования является программа Content (см. ftp://ftp.cwi.nl/pub/CONTENT/). Она может анализировать отображения, системы обыкновенных дифференциальных уравнений и уравнения в частных производных. Программа позволяет не только непосредственно решать систему и выводить в графическом или текстовом виде решение на экран, но и проводить бифуркационный анализ динамической системы. Коротко перечислим типы обнаруживаемых пакетом бифуркаций: ветвление равновесий, бифуркации Андронова–Хопфа, Богданова–Такенса, возникновения инвариантного тора из периодического режима и др.

Программа обладает дружественным интерфейсом, реализует двумерную и трехмерную графику и имеет большие возможности для ввода и вывода результатов. Существуют версии пакета для различных платформ (Linux, SUN OS, Windows и другие). Для работы с пакетом требуется компилятор, например, версия для Windows предполагает наличие у пользователя транслятора Borland C++5.0. Для установки программы требуется получить архив с файлами, развернуть его и запустить инсталляционную программу. Пакет снабжен подробной инструкцией на английском языке. На рис. 21.6 даны окна программы при исследовании системы обыкновенных дифференциальных уравнений.

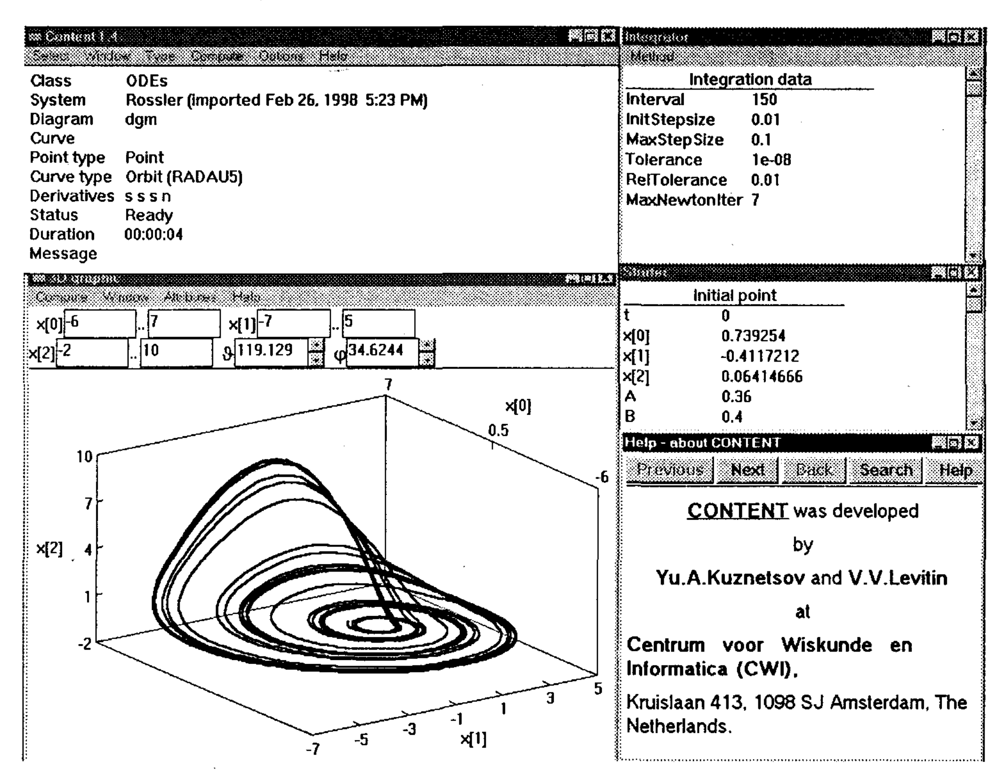


Рис. 21.6. Окна программы Content

Во многом похожими, как по возможностям, так и по интерфейсу, на Content являются программы AUTO (http://indy.cs.concordia.ca/auto/) и DsTool (http://mathlab.cit.cornell.edu/dyn_sys/dstool/dstool.html). Перечисленные программы обладают большими возможностями, но для их квалифицированного использования необходимы серьезные знания теории динамических систем, что иногда делает их использование затруднительным.

Еще одной программой для анализа динамических систем является пакет DESIR. Эта программа была создана в Ростовском университете одним из авторов данной книги В. Н. Говорухиным, и найти ее можно по адресу http://www.math.rsu.ru/mexmat/kvm/MME/desir_r.html.

Программа реализована в виде библиотек для языка Turbo-Pascal и снабжена инструкцией на русском языке. Коротко перечислим основные возможности программы: выбор численного метода решения и анализа динамической системы, поиск точек равновесий, периодических решений (циклов), движение по одному из параметров с поиском точек бифуркаций равновесий и циклов, построение отображения Пуанкаре, вычисление показателей Ляпунова, числа вращения квазипериодических режимов, изображение векторного поля и др. Отметим, что в отличие от перечисленных ранее пакетов DESIR не включает детального бифуркационного анализа. На рис. 21.7 дано окно программы с изображенным странным аттрактором.

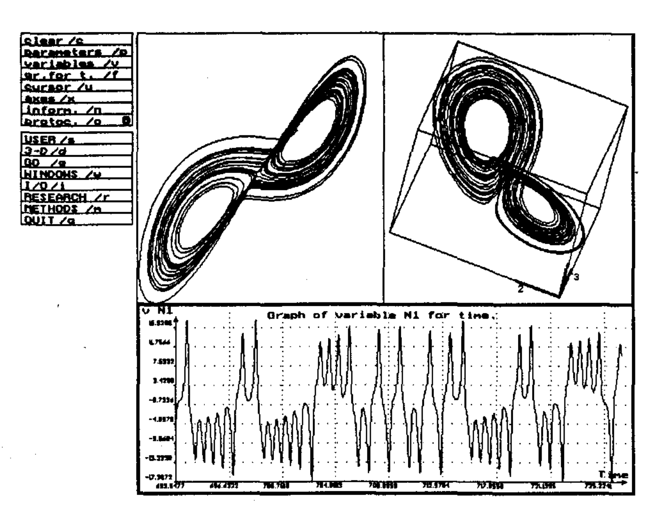


Рис. 21.7. Окно программы DESIR

Рис. 21.8. Окно программы Dynamics Solver

Для ознакомления с основами теории динамических систем и исследования динамики, порождаемой отображениями и обыкновенными дифференциальными уравнениями, замечательным инструментом является программа Dynamics Solver (http://tp.lc.ehu.es/JMA/ds/ds.html). Она обладает дружественным и прозрачным интерфейсом, позволяет наглядно представить результаты прямого счета, но не включа-

ет достаточного числа инструментов качественного анализа. Несомненным достоинством инсталляционного набора файлов является большой архив демонстраций, основанных на классических примерах из области механики, дифференциальных уравнений и хаотической динамики. Окно программы дано на рис. 21.8.

Информационные ресурсы

Здесь мы перечислим сайты, содержащие полезную информацию, необходимую в научной работе и обучении. Как и в предыдущих разделах этой главы, мы ограничимся наиболее полными и постоянными ресурсами.

Общая информация

Самую разнообразную информацию можно найти на сайтах различных обществ и объединений:

- Российская Академия наук: http://www.ras.ru/NIS/nis-emir-n.html;
- Американское математическое общество: http://www.ams.org;
- Европейское математическое общество: http://www.emis.de;
- Европейское физическое общество: http://www.eps.org;
- Американское физическое общество: http://www.aps.org;
- Американский институт физики: http://www.aip.org.

Фонды поддержки научных исследований:

- Российский фонд фундаментальных исследований: http://www.rfbr.ru;
- Фонд «Университеты России»: http://www.uniros.ru;
- Американский фонд CRDF содействия научному сотрудничеству со странами СНГ: http://www.crdf.org;
- Европейский фонд INTAS содействия научному сотрудничеству со странами СНГ: http://www.intas.be/mainfs.htm.

Новости об исследованиях и событиях в области вычислительной математики можно регулярно получать на свой электронный почтовый адрес, подписавшись на дайджесты численного анализа (http://www.netlib.org/na-digest-html) и немецкого общества научных вычислений (http://www.scicomp.uni-erlangen.de).

Конференции

Объявления о проводящихся конференциях в области математики и ее приложений можно найти на страницах перечисленных выше обществ, а также они размещаются по следующим адресам:

- База данных о конференциях Netlib: http://www.netlib.org/confdb/Conferences.html;
- База данных Atlas: http://at.yorku.ca/amca.

Список конференций в области физики можно посмотреть на странице Европейского физического общества: http://epswww.epfl.ch/ephconf.html.

Поиск и просмотр математической литературы

Начнем этот раздел с описания электронных вариантов реферативных журналов, содержащих краткие аннотации статей практически всех математических журналов и изданий. Одним из наиболее полных таких ресурсов является сайт Американского математического общества http://www.ams.org/mathscinet. Однако пользование этой базой является платным и доступно далеко не всем. Аналогичной по количеству материала и сервисным возможностям является база данных MATH database Европейского математического общества, размещенная по адресу http://www.emis.de/ZMATH/. Вообще говоря, она также является платной, но для не зарегистрированных пользователей предусмотрен демонстрационный вариант с усеченными возможностями (на запрос дается только три ссылки). Коротко остановимся на работе с этой базой данных.

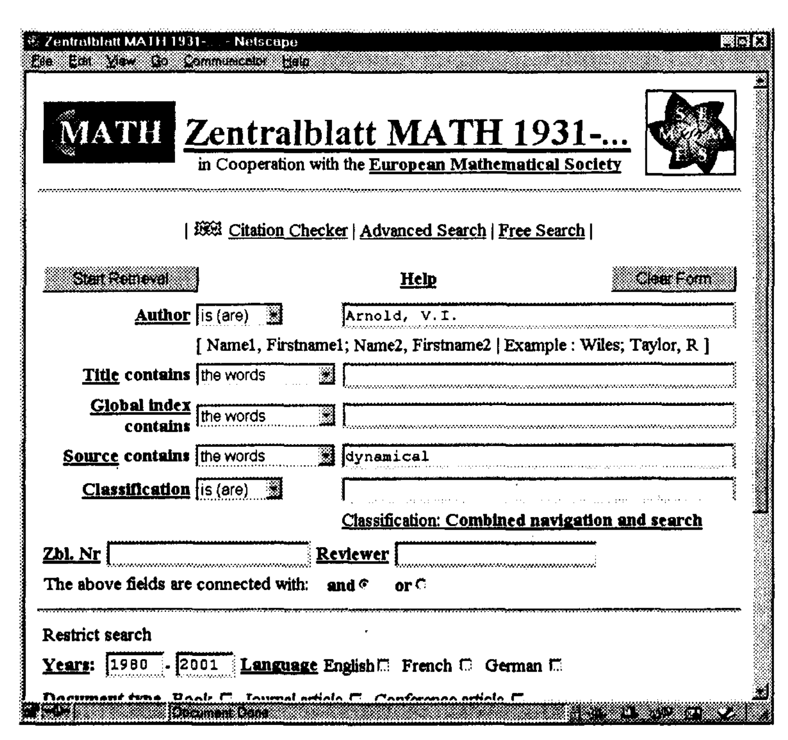


Рис. 21.9. *Окно для запроса поиска рефератов в базе MATH database*

С главной страницы сайта можно обратиться к трем вариантам поиска: Standard Query Form (стандартный поиск), Advanced Query Form (расширенный поиск), Free Query Form (поиск с использованием командного языка). Рассмотрим стандартный поиск. На рис. 21.9 изображено окно для запроса поиска реферата. Предусмотрен поиск по фамилии автора (поле Author), по словам в заголовке статьи или книги (Title), по словам во всех разделах (Global index), по словам в тексте реферата (Source), по кодам классификатора научных областей (Classification), по номеру реферата и фамилии референта (Zbl.Nr. и Reviewer). Существуют способы

сужения поиска с помощью указания года выхода работы, языка и типа издания (книга, журнал, материалы конференции). Понятно, что, сужая область поиска, зачастую вполне можно обойтись демонстрационной версией для полноценной работы с этой базой данных.

После того как нужные поля в странице запроса заполнены (на рис. 21.9 приведен пример этой страницы), следует нажать клавишу Start Retrieval. Результатом выполнения будет страница с найденными ссылками. Например, на рис. 21.10 приведен результат поиска статей автора Arnold, V. I., содержащих в реферате слово dynamical и вышедших в 1980–2001 годах. Так как использовалась демонстрационная версия базы данных, то были получены только три ссылки из пяти. Для просмотра реферата интересующие источники в списке нужно отметить (щелкнуть мышью на квадрате справа от номера источника) или выбрать режим вывода рефератов всех показанных статей (пункт all these entries) и нажать кнопку Display. По умолчанию текст реферата будет выведен на экран в виде HTML-документа, но его можно получить и в виде обычного текста, как текст на языке TeX, в виде файла в формате PostScript или PDF и др. Для этого, перед нажатием кнопки Display, нужно выбрать требуемый формат из списка справа от нее. В зависимости от указанного формата на экране появится или результат, или запрос об имени файла для его сохранения.

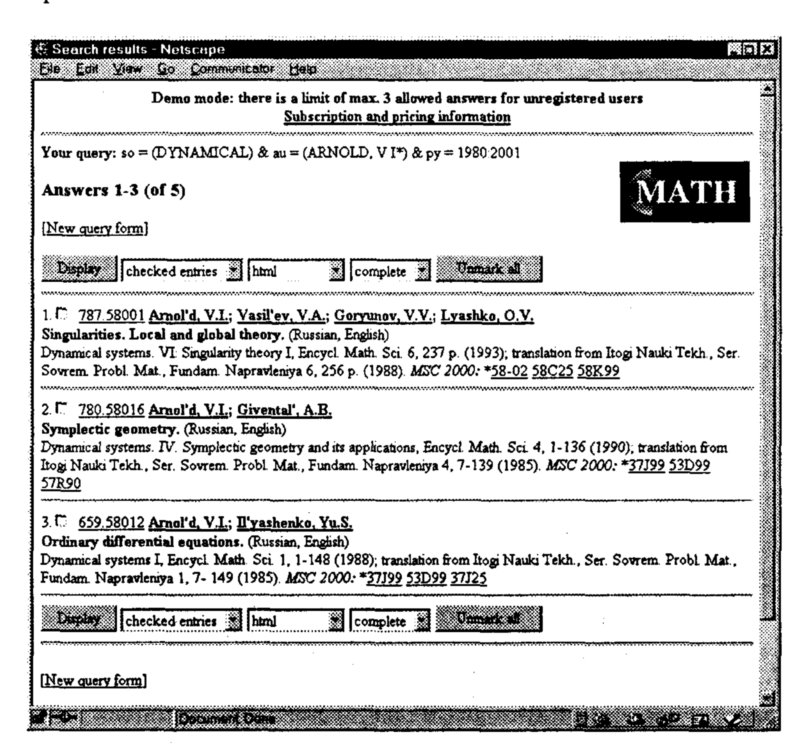


Рис. 21.10. Результат поиска в базе данных MATH database.

Кроме описанной базы MATH database Европейским математическим обществом поддерживается база данных о публикациях, посвященных преподаванию матема-

тики (http://www.emis.de/MATH/DI.html), и архив препринтов (http://euler.zblmath.fiz-karlsruhe.de/MPRESS). В архиве препринтов собраны электронные версии работ из разных областей математики, вышедшие в различных научных центрах и университетах.

Многие журналы, как в нашей стране, так и за рубежом, поддерживают электронные версии. Обычно их просмотр является платным, но свободно можно посмотреть их оглавления, аннотации, и часто существует демонстрационный бесплатный набор статей. Так, на сайте издательства «Наука» (http://www.maik.ru/) можно просмотреть оглавления всех журналов, издаваемых им, аннотации статей и бесплатно предоставляется одна статья любому пользователю, корректно заполнившему бланк заказа. При поддержке Российского фонда фундаментальных исследований развивается «Научная электронная библиотека» (http://www.elibrary.ru). Доступ к ней имеют многие российские университеты и институты. На указанном выше сайте можно найти полные электронные версии отечественных и зарубежных журналов, электронные книги, без ограничений воспользоваться рядом баз данных, в частности MATH database.

В настоящее время большинство университетов и институтов развивают собственные базы данных и коллекции препринтов в электронном виде. Поэтому для поиска книг или статей конкретного автора можно рекомендовать найти в Интернете страницу его места работы и попытаться разыскать там нужный документ.

Большое количество учебных материалов размещается непосредственно самими авторами. Для их поиска можно посоветовать обратиться к поисковым системам (например, http://www.yandex.ru, http://www.altavista.com) или к таким тематическим сайтам, как http://www.exponenta.ru или http://www.referat.ru.

Математические документы в Интернете

В принципе, в предыдущих разделах третьей части книги уже изложено достаточное количество информации для создания различных электронных версий математических документов. Наиболее распространены два способа размещения электронных версий документов в Интернете: в виде файлов и в виде гипертекстовых документов.

В первом случае необходимо подготовить вариант документа в одном из общепринятых форматов. Де-факто на данный момент такими форматами являются PostScript и PDF. Такой стандарт объясняется аппаратной независимостью получающихся файлов и доступностью программного обеспечения для их обработки. В предыдущей главе были описаны сами эти форматы и способы преобразования в них различных документов. Естественно, мы не будем в этой книге обсуждать технику физического размещения файлов в Интернете, обычно этим занимается системный администратор.

Гипертекстовый вариант статьи значительно проще читать с монитора, используя стандартные средства, однако для математических текстов пока не существует универсальных способов отображения формул. Существуют программы просмотра математических текстов, рассчитанные на использование специального языка для

описания формул, однако они не достаточно широко распространены. В настоящее время лучше подготавливать гипертекстовые математические тексты в формате HTML в расчете на такие универсальные браузеры, как Netscape Navigator или MS Explorer.

Как подготовить HTML-версию статьи

HTML-формат является стандартом для представления информации в Интернете и поддерживается всеми броузерами. Команды языка HTML задают правила, по которым браузер выводит гипертекстовый документ на экран: размещение текста в окне, представление графических (рисунков) и других объектов. Вообще говоря, для создания гипертекстовых документов не обязательно знать язык HTML, так как существует ряд способов автоматического преобразования документов различных форматов в HTML. Так, в первой части книги обсуждалась конвертация рабочих документов Maple в этот формат, а в предыдущей главе был описан ряд программ для преобразования файлов на языке LaTeX в HTML-формат. Отметим еще, что все современные офисные приложения, например, Microsoft Word, способны генерировать размеченные в HTML документы из «родного» формата. Кроме того, такие программы, как MS FrontPage, Netscape Composer и другие позволяют создавать страницы в интерактивном режиме. Несмотря на это, мы считаем, что элементарные знания о языке гипертекстовых документов необходимы для их квалифицированного создания.

Краткое введение в язык HTML

Язык HTML (HyperText Markup Language) позволяет формировать самую разнообразную графическую и текстовую информацию. Он содержит команды, позволяющие управлять видом шрифтов, размером и расположением иллюстраций, позволяет осуществлять переход от фрагмента текста или иллюстрации к другим HTML-документом — так называемую гипертекстовую ссылку. Документ в формате HTML представляет собой текстовый файл в формате ASCI, содержащий все необходимые сведения о выводимой на экран информации, и его можно редактировать любым текстовым редактором. Однако при создании документов на русском языке следует помнить о существовании нескольких кодировок символов кириллицы. Бесспорное достоинство HTML заключается в том, что документ в этом формате может быть просмотрен программами различных типов и на различных платформах.

Команды языка HTML (теги) отличаются от команд большинства языков программирования. Эти команды заключаются в треугольные скобки из знаков «больше» и «меньше» и состоят, как правило, из двух частей, между которыми располагается выполняемый командой текст. Такие теги называются парными. Закрывающий фрагмент команды отличается от открывающего наличием символа «/», расположенного перед именем тега. Существуют и непарные теги, например горизонтальная линия задается командой <HR>. Прописные и строчные буквы в именах тегов не различаются.

Каждый HTML-документ начинается тегом <HTML>, который идентифицирует его тип, а заканчивается тегом </HTML>. После тега <HTML> следует заголовок до-

кумента, открывающийся тегом <HEAD> (а закрывающийся в соответствии с правилами языка командой </HEAD>). Внутри заголовка можно с помощью команды <TITLE> *текст* </TITLE> определить текст, который будет возникать в верхней части окна браузера, указать кодировку документа и многое другое.

После заголовка размещается текст документа, который заключается между тегами <BODY> и </BODY>. Команда <BODY> может также включать атрибуты оформления документа — цвет текста, цвет фона, цвета гипертекстовых ссылок, а также имя графического файла, служащего фоном документа. В качестве значения цвета в HTML-формате используются составные шестнадцатеричные комбинации, начинающиеся с символа #, после которого идут три парных группы значений интенсивности каждой цветовой составляющей (RGB). Мы не будем перечислять все возможные атрибуты, а только используем их ниже в примере документа.

HTML-документ структурно делится на обычный текст, заголовки различных уровней, списки, цитаты и другие объекты. Заголовок самого верхнего уровня имеет признак «1», его синтаксис следующий: <H1> *Заголовок первого уровня* </H1>. Синтаксис заголовков более низкого уровня отличается только номером, причем браузеры обычно поддерживают шесть видов заголовков.

Списки бывают упорядоченные (нумерованные) и неупорядоченные (маркированные) и оформляются, соответственно, в виде парных тегов <OL> и <UL>. Элементы списка в обоих случаях определяются парным тегом <LI> (см. пример далее). Цитаты можно задать при помощи тегов <BLOCKQUOTE> и <CITE>, причем в первом случае заключенный между открывающим и закрывающим тегами текст будет выделен отступом, а во втором — курсивом.

Язык HTML включает и некоторые возможности управления форматированием теста. Для центрирования текста можно использовать тег <CENTER> (то есть все элементы между <CENTER> и </CENTER> будут находиться в центре окна). Для перехода к новому абзацу используется непарный тег
. С помощью парных тегов <B>, <I> и <U> можно выделять текст соответственно полужирным шрифтом, курсивом, подчеркиванием. Команда <FONT> позволяет управлять параметрами шрифта, причем она должна обязательно содержать один из атрибутов COLOR=, FACE= или SIZE= (см. пример).

В тексте HTML-документа можно размещать гиперссылки, то есть фрагменты текста, которые являются ссылками на другой документ или файл. Гиперссылка позволяет легко переходить от одного документа к другому по нажатию кнопки мыши. Она определяется при помощи тега <A> с атрибутом HREF, который указывает нужную ссылку, после которой следует текст ссылки. Ссылки могут быть абсолютными (адрес в Интернете) и относительными, то есть указывающими путь к нужному файлу непосредственно на сервере, где размешена страница.

HTML-документ может включать графические иллюстрации, которые должны быть подготовлены в виде файлов форматов GIF и JPG. Рисунки хранятся на сервере в виде отдельных файлов, но изображаются в документе при его просмотре. Для вставки рисунка используется непарный тег <IMG> с обязательным атрибутом SRC, указывающим абсолютную или относительную ссылку на файл с рисунком. По умолчанию при просмотре используются реальные размеры рисунка, но с помощью атрибутов WIDTH= (ширина рисунка в пиксеах) и HEIGHT= (высота рисунка в пикселах) команды <IMG> его можно масштабировать.

Приведем пример документа, иллюстрирующий некоторые перечисленные выше возможности языка HTML. Вид этого документа в окне браузера Microsoft Explorer дан на рис. 21.11:

```
<HTML>
<HEAD> <TITLE> Пример HTML-документа </TITLE>
<META charset=Windows-1251"> </HEAD>
<BODY BGCOLOR=#AAAAAA LINK=#0000FF TEXT=#000000>
<P> <H1> В данной книге описаны: </H1>
<OL>
<LI> <H2> Пакет символьных вычислений  Maple </H2>
<LI> <H3> Пакет численного анализа
        <A HREF="http://www.mathworks.com"> Matlab </A>
        <IMG SRC="MATLAB.GIF"> </H3>
<LI> Набор текстов в системе <B><I> LaTeX </B></I>
   и математические публикации в <FONT size=6> Интернет
</FONT> </LI> </OL>
<HR> <CENTER> <FONT size=2>
Авторы книги будут благодарны читателям за любые комментарии, которые можно направлять
Говорухину В.Н. или Цибулину В.Г. соответственно по адресам:
</FONT> <HR>
<I> vgov@math.rsu.ru </I> <P> <B> tsybulin@math.rsu.ru </B>
</CENTER> </BODY>
</HTML>
```

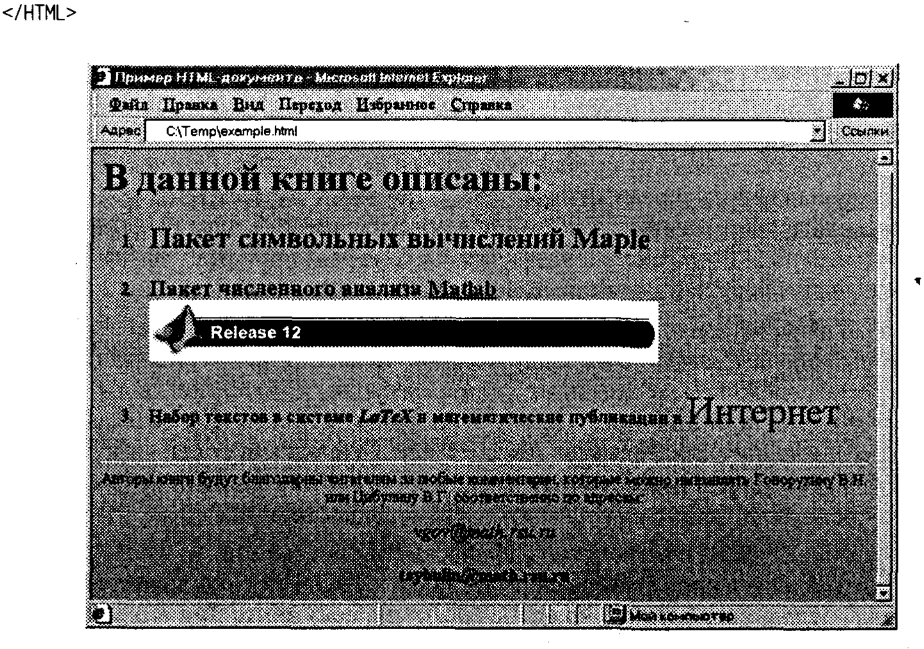


Рис. 21.11. Результат просмотра примера документа в формате HTML

Кроме того, в документе можно создавать таблицы, разбивать HTML-страницу на так называемые фреймы, под которыми понимаются области гипертекстового документа со своими полосами прокрутки, и многое другое. Понятно, что мы дали

очень беглое описание небольшого подмножества команд языка HTML. За подробной информацией следует обращаться к соответствующей литературе или к многочисленным публикациям на эту тему в Интернете. Заметим также, что использование программ, которые называются CGI Script, появление таких технологий, как Java, ActiveX изменили представление о HTML-документе как о тексте со встроенными рисунками и ссылками. Новые технологии позволяют создавать активные документы и использовать при этом все возможности программирования.

Список литературы

Maple

1. *Аладьев В. З., Шишаков М. Л.* Автоматизированное рабочее место математика. Лаборатория базовых знаний, 2000.
2. *Говорухин В. Н., Цибулин В. Г.* Введение в Maple. Математический пакет для всех. М.: Мир, 1997.
3. *Дьяконов В.* Математическая система Maple V. Солон, 1998.
4. *Манзон Б. М.* Maple V Power Edition. М.: Филинъ, 1998.
5. *Матросов А.* Maple 6. Решение задач высшей математики и механики. БХВ-Петербург, 2001.
6. *Прохоров Г., Колбеев В., Желнов К., Леденев М.* Математический пакет Maple V Release 4: Руководство пользователя. Калуга: Облиздат, 1998.
7. *Heal K. M., Hansen M. L., Rickard K. M.* Maple 6. Learning guide. Waterloo Maple Inc. 2000.
8. *Heck A.* Introduction to Maple. Second edition. Springer-Verlag, 1996.
9. *Monagan M. B., Geddes K. O., Heal K. M., Labahn G., Vorkoetter S. M., McCarron J.* Maple 6. Programming guide. Waterloo Maple Inc. 2000.
10. *Redfern D.* The Maple Handbook. Springer-Verlag, 1996.

MATLAB

11. *Гультяев А.* Визуальное моделирование в среде MATLAB: Учебный курс. СПб.: Питер, 2000.
12. *Дьяконов В. П.* Справочник по применению системы PC MATLAB. М.: Физматлит, 1993.
13. *Дьяконов В. П., Абраменкова И. В.* MATLAB 5.0/5.3 Система символьной математики. М.: Нолидж, 1999.
14. *Дьяконов В. П.* MATLAB: Учебный курс. СПб.: Питер, 2000.
15. *Дьяконов В. П., Круглов В.* Математические пакеты расширения MATLAB: Специальный справочник. СПб.: Питер, 2001.
16. *Егоренков Д. Л., Фрадков А. Л., Харламов В. Ю.* Основы математического моделирования. БГТУ. СПб., 1996.

17. *Мартынов Н. Н., Иванов А. П.* MATLAB 5.x. Вычисления, визуализация, программирование. М.: КУДИЦ-ОБРАЗ, 2000.
18. *Потемкин В. Г.* Система MATLAB. М.: Диалог-МИФИ, 1997.
19. *Потемкин В. Г.* MATLAB 5 для студентов. М.: Диалог-МИФИ, 1998.
20. *Потемкин В. Г.* Система инженерных и научных расчетов MATLAB 5.x. В 2-х т. М.: Диалог-МИФИ, 1999.
21. *Using* MATLAB. The MathWorks Inc., 2000.
22. *Using* MATLAB Graphics. The MathWorks Inc., 2000.
23. *Using* MATLAB. The MathWorks Inc., 2000.
24. *Getting* Started with MATLAB. The MathWorks Inc., 1998.
25. *MATLAB* Notebook User's Guide. The MathWorks Inc., 1998.
26. *Building* GUIs with MATLAB. The MathWorks Inc., 1997.
27. *MATLAB* Application Program Interface Guide. The MathWorks Inc., 1998.
28. *Symbolic* Math Toolbox. User's Guide. The MathWorks Inc., 1997.

LaTeX и другие программы

29. *Богумирский Б.* Энциклопедия Windows 98. СПб.: Питер, 2001.
30. *Гуссенс М.: Миттельбах Ф., Самарин А.* Путеводитель по пакету LaTeX и его расширению LaTeX 2ε. М.: Мир, 1999 г.
31. *Дэвис Б., Малдер С., Роуз К.* Photoshop 4-5: учебный курс (+CD). СПб.: Питер-пресс, 1998.
32. *Котельников И., Чеботарев П.* Издательская система LaTeX2ε. Новосибирск: Сибирский хронограф, 1998.
33. *Львовский С.* Набор и верстка в пакете LaTeX. М.: Космосинформ, 1995.
34. *Рабин Ч.* Эффективная работа с Microsoft Word 2000. СПб.: Питер, 2001.
35. *Рихтер Дж.* Windows для профессионалов. СПб.: Питер, 2001.
36. *Adobe* Systems, Inc. PostScript Language. Tutorial and Cookbook. Addison-Wesley Publishing Co. MA 1986.
37. *Adobe* Systems, Inc. PostScript Language Reference Manual. Second Edition. Addison-Wesley Publishing Co. MA 1990.
38. *Knuth D. E.* The TeX book. Volume A of Computers and Typesetting, Addison-Wesley Publishing Company 1984. (Русский перевод: Кнут Д. Е. Все про TeX. Протвино, 1993.)
39. *Lamport L.* LaTeX: A Document Preparation System. Addison-Wesley, Reading, Massachusetts, second edition, 1994.
40. *Oetiker T., Partl H., Hyna I. and Schlegl E.* The Not So Short Introduction to LaTeX 2ε. (CTAN: /tex-archive/info/lshort).

Учебная и научная литература

41. *Акритас А.* Основы компьютерной алгебры с приложениями. М.: Мир, 1994.
42. *Ахромеева Т. С., Курдюмов С. П., Малинецкий Г. Г., Самарский А. А.* Нестационарные диссипативные структуры и диффузионный хаос. М.: Наука, 1992.
43. *Бабенко К. И.* Основы численного анализа. М.: Наука, 1986.
44. *Бахвалов Н. С.* Численные методы. Т.1. М.: Наука, 1975.
45. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* Численные методы. М.: Наука, 1987.

46. *Беллман Р.* Введение в теорию матриц. М.: Наука, 1969.
47. *Воеводин В. В.* Вычислительные методы линейной алгебры. М.: Наука, 1977.
48. *Говорухин В. Н., Моргулис А. Б., Тютюнов Ю. В.* Медленный таксис в модели хищник-жертва // Доклады РАН. 2000. Т. 372. № 6. 2000. С. 730 –732.
49. *Гулин А. В., Самарский А. А.* Численные методы. М.: Наука, 1989.
50. *Демидович Б. П.* Сборник задач и упражнений по математическому анализу. М.: Наука, 1977.
51. *Задачи* и упражнения по математическому анализу для втузов / Под ред. Б. П. Демидовича. М.: Наука, 1970.
52. *Калиткин Н. Н.* Численные методы. М.: Наука, 1978.
53. *Каханер Д., Моулер К., Нэш С.* Численные методы и программное обеспечение. М.: Мир, 1998.
54. *Клейн Ф.* Лекции о развитии математики в XIX столетии: В 2 т. Т.1. М.: Наука, 1989.
55. *Кнут Д.* Искусство программирования для ЭВМ. Т. 1: Основные алгоритмы. М.: Мир, 1976.
56. *Кнут Д.* Искусство программирования для ЭВМ. Т. 2: Получисленные алгоритмы. М.: Мир, 1977.
57. *Кнут Д.* Искусство программирования для ЭВМ. Т. 3: Сортировка и поиск. М.: Мир, 1978.
58. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2000.
59. *Кудрявцев Л. Д, Кутасов А. Д., Чехлов В. И., Шабунин М. И.* Сборник задач по математическому анализу. М.: Наука, 1986.
60. *Кудрявцев Л. Д.* Математический анализ: В 2 т. М.: Высшая школа, 1970.
61. *Ланда П.С.* Нелинейные колебания и волны. М.: Наука, 1997.
62. *Математическая* энциклопедия. М.: БСЭ.
63. *Марчук Г. И.* Методы вычислительной математики. М.: Наука, 1983.
64. *Моденов П. С., Пархоменко А. С.* Сборник задач по аналитической геометрии. М.: Наука, 1976.
65. *Мун Ф.* Хаотические колебания. М.: Мир, 1990.
66. *Олвер П.* Приложение групп Ли к дифференциальным уравнениям. М.: Мир, 1989.
67. *Самарский А. А.* Теория разностных схем. М.: Наука, 1977.
68. *Самарский А. А.* Введение в численные методы. М.: Наука, 1984.
69. *Фаддеев Д. К., Соминский И. С.* Сборник задач по высшей алгебре. М.: Наука, 1972.
70. *Флетчер К.* Численные методы на основе метода Галеркина. М.: Мир, 1988.
71. *Форсайт Дж., Моулер К.* Численное решение систем линейных алгебраических уравнений. М.: Мир. 1969.
72. *Форсайт Дж., Малкольм М., Моулер К.* Машинные методы математических вычислений. М.: Мир. 1977.
73. *Хайрер Э., Нерсет С., Ваннер Г.* Решение обыкновенных дифференциальных уравнений. М.: Мир. 1986.
74. *Хеминг Р. В.* Численные методы. М.: Наука, 1972.
75. *Шень А.* Программирование: теоремы и задачи. МЦНМО, 1995. (Свободно распространяемый текст книги можно получить в Интернет по адресу: ftp://ftp.mccme.ru/ pub/ users/shen/progbook.)

76. *Эдвардс Р.* Ряды Фурье в современном изложении: В 2 т. М.: Мир, 1985.
77. *Юдович В. И.* Асимптотика предельных циклов системы Лоренца при больших числах Рэлея. Деп. ВИНИТИ N2611-78.
78. *Govorukhin V. N., Morgulis A., Yudovich V. I. and Zaslavsky G. M.* Chaotic advection in compressible helical flow. Physical Review E, 1999, V.60, N 3, pp. 2788-2798.
79. *Lorenz E. N.* On the prevalence of aperiodicity in symple systems. Global Analysis, Calgary, 1978. vol. 755, p.53-75.
80. *Murray J. D.* Mathematical Biology. Springer, New York, 1993.
81. *Volterra V.* Lecons sur la theorie mathematique de la lutte pour la vie. Gauthiers-Villars, Paris, 1931.

Алфавитный указатель

Символы

A

B

C

E

F

G

H

K

L

M

N

O

T

U

V

W

X

Y

Z

А

Б

В

М

Н

О

П

У

Ф

Х-Ц

Ч

Ш

Э

Я